Subject: c64/How to predict when garbage collection occurs..
Posted by Mary Lea Beinhower on Sat, 14 Sep 2013 00:02:58 GMT
View Forum Message <> Reply to Message

Greetings:
With which ZP pointers would I use to detect, beforehand, when the c64 performs it's own Garbage Collection.. Like, would anyone be able to help me write some code I can attach to the IRQ.

When there is only a tiny little bit of memory left, set a address to (1). Then, when I do my own garbage collection, I can NULL that address. All the while, having an IRQ routine that polls the memory location for a (1), if true, let me collection garbage, and zero that address.

If the address is zero, the irq code will exit..

Im trying to NOT let the c64 collect on its own.. force my own collection code.

Thanks In Adavance! :)

--Steve

Subject: Re: c64/How to predict when garbage collection occurs..
Posted by Anton Treuenfels on Sat, 14 Sep 2013 02:53:21 GMT
View Forum Message <> Reply to Message

<mbeinhower@gmail.com> wrote in message
news:95f7b290-79c2-46c6-8860-d8dfd52b74dd@googlegroups.com...
Greetings:
With which ZP pointers would I use to detect, beforehand, when the c64 performs it's own Garbage Collection.. Like, would anyone be able to help me write some code I can attach to the IRQ.

When there is only a tiny little bit of memory left, set a address to (1). Then, when I do my own garbage collection, I can NULL that address. All the while, having an IRQ routine that polls the memory location for a (1), if true, let me collection garbage, and zero that address.

If the address is zero, the irq code will exit..

Im trying to NOT let the c64 collect on its own.. force my own collection code.

Thanks In Adavance! :)

--Steve

=====================================

Garbage collection on the C64 happens when the Bottom-Of-Strings pointer ($33/34) is about to go below the Top-Of-Arrays pointer ($31/32). The BASIC interpreter wants to allocate a dynamic string and finds there is not enough space, so it does a garbage collection and then checks again (if there still isn't enough room, an "Out of Memory" error happens).

So one way to do what you want is to have your IRQ routine check to see if $34 minus $32 is zero or negative, or maybe one or two if you want to be sure to intercept in time.

Another approach might be to note that the BASIC FRE() function also triggers a garbage collection so that it can properly report how much space there is between the BOS pointer and the TOA pointer. Strategically placing FRE() in code so that garbage collection happens at times convenient for a program is sometimes possible. Or you could set up a USR() vector to trigger your routine any time you wanted - why wait for the last minute?

Subject: Re: c64/How to predict when garbage collection occurs..
Posted by Christian Brandt on Sat, 14 Sep 2013 12:15:29 GMT

View Forum Message <> Reply to Message

On 14.09.2013 04:53, Anton Treuenfels wrote:

> Another approach might be to note that the BASIC FRE() function also
> triggers a garbage collection so that it can properly report how much
> space there is between the BOS pointer and the TOA pointer.
> Strategically placing FRE() in code so that garbage collection happens
> at times convenient for a program is sometimes possible. Or you could
> set up a USR() vector to trigger your routine any time you wanted - why
> wait for the last minute?

Another approach: Replace the garbage handler completely.

Deep in my memories there was an article in the german 64er magazine in the late 80ths doing exactly this. The reworked garbage collection was A LOT faster (100 times at least) and also offered some feed back for checking current state and starting different types of garbage collection (fast, normal and slow). Also I remember the whole handler was expanded later with an interesting option of using $A000-$FFFF-RAM for arrays - yes really only arrays and all arrays always went to $a000-$ffff at cost of some performance, effectively expanding basic memory from 39k bytes to some 60k.

--
Christian Brandt

life is short and in most cases it ends with death
but my tombstone will carry the hiscore

Subject: Re: c64/How to predict when garbage collection occurs..
Posted by Mary Lea Beinhower on Sat, 14 Sep 2013 19:18:57 GMT
View Forum Message <> Reply to Message

On Friday, September 13, 2013 8:02:58 PM UTC-4, mbein...@gmail.com wrote:
> Greetings:
>
> With which ZP pointers would I use to detect, beforehand, when the c64 performs it's own Garbage Collection.. Like, would anyone be able to help me write some code I can attach to the IRQ.
>
>
>
> When there is only a tiny little bit of memory left, set a address to (1). Then, when I do my own garbage collection, I can NULL that address. All the while, having an IRQ routine that polls the memory location for a (1), if true, let me collection garbage, and zero that address.
>
>
>
> If the address is zero, the irq code will exit..
>
>
>
> Im trying to NOT let the c64 collect on its own.. force my own collection code.
>
>
>
> Thanks In Adavance! :)
>
>
>
> --Steve

I appreciate the input. And yes, the code Christian is talking about is called, 'The Sanitation Engineer" and is 100% amazing, except the program im writing, uses memory under basic ROM ($A000 - $BFFF).

Anton:
Do you think you could.. somehow.. come-up with a (Holier than Gawd) code snippet, just to see, exactly, how you would approach this scenario??

If not, ill still hammer away at it, till I get it right.

(*thank goodness for today's cross compiling setups*)

-Regards,
--Steve

Subject: Re: c64/How to predict when garbage collection occurs..
Posted by Anton Treuenfels on Sun, 15 Sep 2013 02:43:48 GMT
View Forum Message <> Reply to Message

Anton:
Do you think you could.. somehow.. come-up with a (Holier than Gawd) code snippet, just to see, exactly, how you would approach this scenario??

Uh, I'm not sure I'd try approaching the problem from the direction of an IRQ in any case. I'm not quite sure if you're trying to monitor the situation from BASIC via a flag set by an IRQ, or trying to do a complete garbage collection within an IRQ handler, or both.

In the first case you'd have to poll the flag from within BASIC, presumably using PEEK. If you can do that regularly enough to matter then you could also get effectively the same result by PEEKing the locations of interest and subtracting their values directly without needing to fiddle with an IRQ. As a numerical operation doing so would not trigger a garbage collection unless you decided to perform one as a result.

In the second case, no matter how much you managed to speed up the process it would still be too long for a single interrupt.

In the third case (ha! you didn't know about that one, did you?!) it's all moot unless you have a faster garbage collector to use instead of the built-in one. Have you?

I did write a replacement once. It worked, but perhaps the only elegant thing about it was that I tied it to the FRE() function so it was fairly transparent to the system. Other than that it was quite an ugly hack.

Subject: Re: c64/How to predict when garbage collection occurs..
Posted by Dombo on Sun, 15 Sep 2013 09:34:37 GMT
View Forum Message <> Reply to Message

Op 15-Sep-13 4:43, Anton Treuenfels schreef:
>
> Anton:

> Do you think you could.. somehow.. come-up with a (Holier than Gawd)
> code snippet, just to see, exactly, how you would approach this scenario??
>
> ---------------------------
>
> Uh, I'm not sure I'd try approaching the problem from the direction of
> an IRQ in any case. I'm not quite sure if you're trying to monitor the
> situation from BASIC via a flag set by an IRQ, or trying to do a
> complete garbage collection within an IRQ handler, or both.
>
> In the first case you'd have to poll the flag from within BASIC,
> presumably using PEEK. If you can do that regularly enough to matter
> then you could also get effectively the same result by PEEKing the
> locations of interest and subtracting their values directly without
> needing to fiddle with an IRQ. As a numerical operation doing so would
> not trigger a garbage collection unless you decided to perform one as a
> result.
>
> In the second case, no matter how much you managed to speed up the
> process it would still be too long for a single interrupt.
>
> In the third case (ha! you didn't know about that one, did you?!) it's
> all moot unless you have a faster garbage collector to use instead of
> the built-in one. Have you?
>
> I did write a replacement once. It worked, but perhaps the only elegant
> thing about it was that I tied it to the FRE() function so it was fairly
> transparent to the system. Other than that it was quite an ugly hack.

Another issue with doing the garbage collection from an IRQ handler is reentrancy.

I doubt the BASIC interpreter, and the garbage collection routine in particular, is written with reentrancy in mind. Imagine the BASIC interpreter loading some pointers to string data, gets interrupted by the IRQ handler which shuffles the location string data. Once the IRQ handler has done its thing the BASIC interpreter continues were it was interrupted, at which point the pointers it loaded before the interrupt occurred are no longer valid. The effect would be that at seemingly random times your BASIC program would act weird or locks up the computer.

Subject: Re: c64/How to predict when garbage collection occurs..
Posted by Anonymous on Sun, 15 Sep 2013 10:45:36 GMT
View Forum Message <> Reply to Message

Originally posted by: Frank Buss

Anton Treuenfels wrote:

> I did write a replacement once. It worked, but perhaps the only elegant
> thing about it was that I tied it to the FRE() function so it was fairly
> transparent to the system. Other than that it was quite an ugly hack.

There is an article in the C64 Wiki has about the Garbage Collection (in German, but the Google translator works fairly well for technical articles)

http://www.c64-wiki.de/index.php/Garbage_Collection

The article shows an example, which causes the GC to run 1 hour and 49 minutes. One of the three better implementations mentioned at the bottom of the article runs the same in less than 4 seconds.

I would just call this new GC every now and then. No need to change the FRE function itself.

BTW: currently there is a GC competition running:

http://www.forum64.de/wbb3/board25-coder-unter-sich/board308
-programmieren/board26-basic/54109-basic-garbage-collection- compo/

The goal is to write a Basic program with the longest GC run. I managed already infinite, so would be hard to beat :-)

--
Frank Buss, http://www.frank-buss.de
electronics and more: http://www.youtube.com/user/frankbuss

Subject: Re: c64/How to predict when garbage collection occurs..
Posted by Anton Treuenfels on Mon, 16 Sep 2013 04:28:44 GMT
View Forum Message <> Reply to Message

"Frank Buss" <fb@frank-buss.de> wrote in message
news:l1434h$esa$1@newsreader4.netcologne.de...

> BTW: currently there is a GC competition running:
>
> http://www.forum64.de/wbb3/board25-coder-unter-sich/board308
> -programmieren/board26-basic/54109-basic-garbage-collection- compo/
>
> The goal is to write a Basic program with the longest GC run. I managed
> already infinite, so would be hard to beat :-)

Then I imagine the next step is to write the *shortest* program that causes the GC to run for an infinite amount of time :-)

Subject: Re: c64/How to predict when garbage collection occurs..
Posted by epc8 on Tue, 22 Oct 2013 05:59:05 GMT

View Forum Message <> Reply to Message

On Saturday, September 14, 2013 8:15:29 AM UTC-4, Christian Brandt wrote:
> On 14.09.2013 04:53, Anton Treuenfels wrote:
>> Another approach might be to note that the BASIC FRE() function also
>> triggers a garbage collection so that it can properly report how much
>> space there is between the BOS pointer and the TOA pointer.
>> Strategically placing FRE() in code so that garbage collection happens
>> at times convenient for a program is sometimes possible. Or you could
>> set up a USR() vector to trigger your routine any time you wanted - why
>> wait for the last minute?

> Another approach: Replace the garbage handler completely.

> Deep in my memories there was an article in the german 64er magazine in
> the late 80ths doing exactly this. The reworked garbage collection was A
> LOT faster (100 times at least) and also offered some feed back for
> checking current state and starting different types of garbage
> collection (fast, normal and slow).

This problem was also well studied on the Apple 2. ISTR articles from both "Call APPLE" and "Micro" from the early 80s written by Cornelis Bongers.

See this reference for some explanation and code in a different publication.

http://www.txbobsc.com/aal/1984/aal8403.html

Reassembling this for CBM might work with minor changes to ORG, z-page locations and internal entry points.

(I hang out sometimes in CSA2, but rarely here.)