Subject: is memsetex really at optimal speed?
Posted by piotr5 on Tue, 19 Dec 2023 01:18:07 GMT
View Forum Message <> Reply to Message

it seems this function just repeats copying the object piece by piece. on the plus-side, that's quite "atomic", no object remains half-copied for long. but I'm not sure the speed really scales for initializing larger data-collections.

what I imagined it should do is something like this (I'm hereby giving permission to use the code below):

```
inline
void memsetyx(void *t, const void *item, int item_size, int count) {
 ASSERT(item_size >= 0);
 if(count<3||count*item_size<64){memsetex(t,item,item_size,count);return;}
 byte *q = (byte *)t;
 byte *tt=q;
 while(q-tt<64){
  memcpy8(q, item, item_size);
  q+=item_size;
  --count;
 }
 memcpy8(q,tt,qword(count)*item_size);
 memcpy128(tt+item_size*count-16,q-16,1);
}
```

(where the last line could have been avoided if memcpy8__ would perform the Copy128(len - 16); right before the return-statement underneath and at the end.)

haven't tested it though, would that work? is it faster on various platforms? maybe use a bigger constants in the if-statement at the beginning? afaik standard memcpy does allow for source region and destination-region overlapping, am I wrong? admittedly it is rarely needed to initialize an array with lots of copies of complicated objects, but in some prototype-code I could imagine it would happen. in production-code such things likely get optimized out by the programmers though. so this really is not a request to change anything. just asking if that was ever considered and how the discussion went...

Subject: Re: is memsetex really at optimal speed?
Posted by mirek on Thu, 04 Jan 2024 18:25:13 GMT
View Forum Message <> Reply to Message

piotr5 wrote on Tue, 19 December 2023 02:18it seems this function just repeats copying the object piece by piece. on the plus-side, that's quite "atomic", no object remains half-copied for long. but I'm not sure the speed really scales for initializing larger data-collections.

what I imagined it should do is something like this (I'm hereby giving permission to use the code below):

```
inline
```

```
void memsetyx(void *t, const void *item, int item_size, int count) {
 ASSERT(item_size >= 0);
 if(count<3||count*item_size<64){memsetex(t,item,item_size,count);return;}
 byte *q = (byte *)t;
 byte *tt=q;
 while(q-tt<64){
  memcpy8(q, item, item_size);
  q+=item_size;
  --count;
 }
 memcpy8(q,tt,qword(count)*item_size);
 memcpy128(tt+item_size*count-16,q-16,1);
}
```

(where the last line could have been avoided if memcpy8__ would perform the Copy128(len - 16); right before the return-statement underneath and at the end.)

haven't tested it though, would that work? is it faster on various platforms? maybe use a bigger constants in the if-statement at the beginning? afaik standard memcpy does allow for source region and destination-region overlapping, am I wrong? admittedly it is rarely needed to initialize an array with lots of copies of complicated objects, but in some prototype-code I could imagine it would happen. in production-code such things likely get optimized out by the programmers though. so this really is not a request to change anything. just asking if that was ever considered and how the discussion went...

You can never tell before you benchmark it... did you?

I remember doing something like that in the past, I believe it was not worth it. I might be wrong....

Mirek
