

Re: number of machine instructions – for algorithm measuring

Source: http://coding.derkeiler.com/Archive/C_CPP/comp.lang.c/2005-02/3156.html

From: ben (*x_at_x.x*)

Date: 02/23/05

Date: Wed, 23 Feb 2005 17:00:05 GMT

In article <1109111407.771992.271560@g14g2000cwa.googlegroups.com>, Mark F. Haigh <mfhaigh@sbcglobal.net> wrote:

```
> ben wrote:
>
> <snip>
>
>>
>>
>> Prove an upper bound on the number of machine instructions required
> to
>> process N objects using the below programme. You may
>> assume, for example, that any C assignment statement always requires
>> less than c instructions, for some fixed constant c.
>>
>> /* a very simple and silly example "algorithm" */
>> #include <stdio.h>
>> #define N 10000
>>
>> int main(void)
>> {
>> int n = 0;
>> int x[3] = { 1, 4, 9 };
>> long unsigned total = 0;
>>
>> while( n < N ) {
>> total += x[ n % 3 ];
>> n++;
>> }
>>
>> printf("%lu\n", total);
>> return 0;
>> }
>>
>
> In my opinion, this is a waste of time. Instruction count no longer
```

comp.lang.c: Re: number of machine instructions – for algorithm measuring

> *has much of a bearing on how much time a program takes to execute. To*
> *use your example program, with N set to 0x4FFFFFFu:*
>
>
> [mark@icepick ~]\$ gcc -save-temps -Wall -O3 -o bar bar.c && time ./bar
> && wc -l bar.s
> 1968526669
>
> real 0m16.683s
> user 0m16.682s
> sys 0m0.002s
> 44 bar.s <- lines of assembly
>
>
> [mark@icepick ~]\$ gcc -save-temps -Wall -O3 -march=pentium4 -o bar
> bar.c && time ./bar && wc -l bar.s
> 1968526669
>
> real 0m4.313s
> user 0m4.312s
> sys 0m0.001s
> 78 bar.s <- lines of assembly
>
>
> *As you can see, a change in an optimization setting roughly doubled the*
> *assembly output, but resulted in a 4x speed improvement. I picked the*
> *quickest of 3 trials each, for the record.*
>
> *It may be an interesting academic exercise, but its real world*
> *applicability is questionable. Probably not the answer you wanted, but*
> *an answer nonetheless.*

Mark,

well, it sounds like a good point. if you're right then it's a very good answer and one i'd definitely want. the exercise is just from a book. i don't have to do it as stated by the exercise. if the way that's detailed isn't so useful and there's a much more useful way then i'd prefer to learn and do it a way that's useful. i do intend to use this stuff in the real world -- that's what i'm learning it for.

i'm just wondering though, if you're comparing operation counts of code compiled using the same settings, and the different programmes are written in a similar fashion (that is, for example, not comparing one programme that unrolls a loop and another that doesn't -- code written in the same style) then the number of operations is a reasonable way to compare? basically, use the same base settings and style. (although "code written in the same style" is probably pretty ambiguous)

further thought: different operations take different amounts of time don't they? -- i think so. so that'd make operation counting less

Re: number of machine instructions – for algorithm measuring

comp.lang.c: Re: number of machine instructions – for algorithm measuring

useful. for instance i think accessing a word of memory that isn't in the cpu's cache can take several clock cycles, whereas if that data is in cache the access might take just one clock cycle. correct?

so basically you're saying timing is the way to measure algorithms?

thanks, ben