

Re: Timing

Source:

<http://www.tech-archive.net/Archive/DotNet/microsoft.public.dotnet.languages.csharp/2004-02/4646.html>

From: Shawn B. (*leabre_at_html.com*)

Date: 02/23/04

Date: Sun, 22 Feb 2004 22:04:01 -0800

What I think I will try to do is this:

Determine how many native CPU cycles lapse during a 1 second period. I currently have a custom timer class that uses the high performance counter API (whatever it is, I'm not looking at it) and is extremely accurate to 7 digits past the decimal in terms of seconds (on my CPU).

Then once I determine how long a simulated cycle should be (based on number of native CPU cycles / number of MHz I wish to simulate). Then each time an instruction is executed it calculates and remembers how long should lapse before the next simulated instruction can execute. Simulated instructions can be 1..7 cycles. Then, when the next instruction wants to execute, it checks to see if the the appropriate time has lapsed before it can execute else it goes into a throttling loop.

Keep in mind, I'm only creating a row processor that does nothing more than simulates the processor. The CPU core has some delegates you can use to make more sense of it. For example, if I want to write a small plugin that detects when a certain memory range has been modified, it can then convert the memory range into some text display to simulate an Apple // screen, for example, or a graphics area, or whatever.

So by the time my CPU instruction has executed, those who are consuming the delegates (whatever the proper term is) can then do thier thing and when all that is done, either there is still time to wait before the next CPU or it took longer and will slow everything down but at least then the next instruction can execute.

So far, I have a simulator work bench that is ideal for the 6502, 65c02, and 65816 that shows the memory map, registers, stack, zero page, a disassembly of the memory. Then, you have to specify what processor to use. The processor is actually a plugin to the simulator environment. The environment can have other plugins that represent an assembler, a debugger, whatever. In my case, I have an assembler (but no linker at the mement).

It sounds like it would slow it down having all these plugins but it's

actually quite fast, I'm impressed at how fast C# can be at this kind of thing. But I still have some optimizations to do. It can be better. Currently, I'm executing at the equivalent of about 433 MHz (simulated) but that's why I need to throttle it, it's too fast.

The debugger, and hence the whole purpose of this project, since everything is done in assembly language for this thing, will allow me to modify the assembly source files while in a debugging breakpoint and the next time I step it will reassemble the program and overlay it back into memory as it is represented in the source files so you can keep making changes while debugging (similar to the VB6 debugger).

So my whole point is this: I want to have at the best, a correctly simulated MHz throttling loop, and at worst, if someone's plugin is too slow, when the next instruction is decoded and executed it'll just execute because there's no need to throttle if it took too long to execute the previous (from its perspective).

I can't use the timer and execute an event at an interval. This is not threaded and I'm dependant on how the plugins perform. So the main loop is actually just a do { } loop that calls CPU.Fetch(); CPU.Execute(); given some other conditions to know when to exit and set a breakpoint and so on.

Anyone interested in the source code when completed?

So far, the 65c02 CPU is written, does everything except the branch instructions, and the debugger is still in progress. I've been working on it part time every day for about 3 weeks. Deciphering the data sheets for the processor is a really tough thing.

Thanks,
Shawn

"Malek" <kemmou@arrabeta.com> wrote in message
news:eXTSCXc%23DHA.3220@TK2MSFTNGP10.phx.gbl...
> *Sorry, I am flatly wrong ... if you do control your cycles, at the speed
you
> are talking about, Richard is totally right ... a timer uses ticks, and
will
> not be slowed down... with time slicing, you can be pretty sure that you
are
> not getting any real delays to your execution..*
>
> "Malek" <kemmou@arrabeta.com> wrote in message
> news:eYuzapc%23DHA.2824@tk2msftngp13.phx.gbl...
> > *using a timer would seem a good option for slowing down the execution,
> > however, the equation 1/speed is not good unless you can consider the
> > execution of your code and anything else happening on the machine to be
> > insignificant ... I would benchmark to see how accurate that assertion
is,
> > and would adjust accordingly... maybe using some performance counters on*

> *the*
> > *process could help on the benchmark or could be added as adjusting*
factor
> > *during execution (to account for whatever else is going on on the*
machine
> > *that might slow your execution)*
> > *"Richard Sadler" <ihaveaholeinmyarm@blueyonder.co.uk> wrote in message*
> > *news:9E95FFFD-3907-43F0-BC7E-EADC55424457@microsoft.com...*
> > > *Hey*
> > >
> > > *You could use a timer (System.Timers.Timer) and set the delay to 1 /*
> *speed*
> > *(so the time is in seconds) and then use the event to perform the next*
> > *instruction on the processor.*
> >
> >
>
>