

Re: "We Never Use Assembly Language"

## Re: "We Never Use Assembly Language"

---

*Source:* <http://coding.derkeiler.com/Archive/Assembler/alt.lang.asm/2006-03/msg00445.html>

---

- *From:* o//annabee <fack@xxxxxxxxxxxxxxxx>
  - *Date:* Sun, 12 Mar 2006 23:39:08 +0100
- 

På Sun, 12 Mar 2006 20:57:29 +0100, skrev nessuno <fmdf66@xxxxxxxx>:

After a long time I haven't been reading from this NG it happens that today I am here watching a few threads and finding that not much has changed with respect to Wannabee way of thinking about programming, especially about efficiency of programming with Assembly.

My efficiency has increased A LOT.

These days I have started to hack some Linux kernel code in the scheduler subsystem because I need some information to be delivered to user space about processes when migrating from one CPU to another in SMP machines.

The code I have to read, study and hack in order to get those statistics is 95% C code (actually GNU/C code). The most of it is located in two files that are, relative to the location of kernel code, `./include/kernel/sched.h` `./kernel/sched.c`. They are about 12,000 lines of C code that I have to read and study.

I's say "wow". It means that I simply will modify two files in order to have my hacks working in a lot of different machines without the need to understand each different CPU's architecture and without the need to be able to write about 12 different Assembly syntaxes. Can you imagine I had to study Assembly for Compaq Alpha AXP, Sun SPARC and UltraSPARC, Motorola 68000, PowerPC, PowerPC64, ARM, Hitachi SuperH, IBM S/390, MIPS, HP PA-RISC, Intel IA-64, DEC VAX, AMD x86-64, AXIS CRIS, and Renesas M32R architectures?

It means that if my work will some day be appreciated by the community of Linux kernel developers I will not be forced to "translate" my code in all those Assemblies in order to have it merged in the official tree.

## Re: "We Never Use Assembly Language"

Ehi, Wannabee? Has it all above mentioned anything to do with productivity of HLL compared to Assembly?

Sounds very impressive, indeed! But what is it that you have done?

Just one more thought. Did I understand well that a RosAsm executable must be produced from a single file?

No. I used a preparser called IncIncluder to help me partition and share my "codebase" between several applications.

Did I understand well that you can't link together different object files to produce a single executable?

Yes, not at the moment, but you can link to a EXE file or a DLL file, or any 1000 other ways to do it. You know, we are asm programmer. We are CREATORS. We are not limited by some cuewords learned from a HLL. We CREATE the High levels we need. It really just funny to listen to the people japping about linking.

Did you know how long it took me to create, independently, a run time length encoder and decoder in asm ? It took about two minutes for the encoder, and a half an hour for the decoder, and it was written on the run, with no plan, while actually writing another routine. This encoder, decoder was only a small piece of a routine to write a file. What a fuck do I need a library to do bitmaps for me, when I can create it in less time than it takes me to even learn how to use the demented library???

The major efficiency of asm, is that you do not need to concern yourself with all the things that you really do not need. Specific. And that converts ALL the time you use to study the documentation written by others into actual devtimes. I have much more time available to do things that I like, that is not related to programming directly, now, than I ever had when using Delphi. And I still work much faster.

If it were applied to Linux kernel code we couldn't have had that beautiful mechanism that is the module loading/unloading that permits to link pieces of code (like device drivers) at run-time into a working kernel. Do you understand what does it mean in terms of memory savings? The kernel is able to link module's code at will, e.g when a new piece of hardware is attached at a running machine, and to unlink the module's code when it doesn't need it anymore.

Great! And when can you free some time at DEVELOPMENT ?

Can you do it with RosAsm, seen that it can't even link different pre-compiled objects together in a single executable? Please forgive me and explain if I got wrong informations about Rosasm.

Re: "We Never Use Assembly Language"

## Re: "We Never Use Assembly Language"

Of course you can do it. Goes without saying. But why would you want to?

For me the answer is. I do not want to. In fact, I would rather be imprisoned into Guantanamo than have to deal with those C shits. :))))

Maybe not, but today, when even thinking about going HLL, my stomach revolts in ways I did not know it was capable of.

Some months ago, when I heard about RosAsm in this NG, I really wanted to give it a try but later I discovered that it can't work on my operating system and I don't want to purchase a MS-Win licence only to run Ros Asm. I hope one day to see it running on a free and open source operating system, no matter which.  
fabio de francesco

Yes. Its pity that Linux cannot offer something like RosAsm. If nobody else will do it, then, sooner or later, most likely later, I will at least `_try_` todo this. As long as Linux does not even able to install on my machine, and as long as ReactOS is not yet able to install at my machine, I can learn as much as possible, in less time, using RosAsm on win32.

I can reveal that I am now rewriting, from scratch, everything, to be as loosely tied to win32 as possible. Only the final steps have to go via the OS. This will be a great help, when I take my `_NEW_` codebase, rewrite it to NASM for some testruns on Linux.

Actually, I am more and less ready todo this now. I may do NASM for windows first, with the current codebase, and then make the final transition to Linux. But development in RosAsm is 100 times faster then in ANY OTHER DEV TOOL. So it will be with RosAsm that I work, and think, and then I just copy to a another assembler.

In all cases, this is the final goal. Either Linux or ReactOS. Or even both. But I am not in any hurry. Theres something called real life, and at the moment its a seriously pain in the butt to me, but say, within a year or two or 3, hopefully :) Who can promise such a thing?

—

Bansai!

.