

Re: Two Click disassembly/reassembly

Source: <http://coding.derkeiler.com/Archive/Assembler/alt.lang.asm/2006-01/msg01413.html>

- *From:* "randyhyde@xxxxxxxxxxxxxx" <randyhyde@xxxxxxxxxxxxxx>
 - *Date:* 24 Jan 2006 09:48:35 -0800
-

Alex McDonald wrote:

>
> OK, how about machines with fewer registers.

Map the extra x86 registers to memory. Accumulator and stack machines (which are about the only machines with fewer registers than the x86) generally have a decent access to memory.

> Flags in an addressable
> register.

What's so hard about that?

> No base/index/offset instructions.

As long as you've got indirection through a register and the ability to add values together, you can certainly simulate this.

> Other than 32 bits.

If the target processor is less than 32-bits, you can use multi-precision arithmetic. If the target processor is greater than 32 bits, you can *usually* ignore the extra bits and do everything with just 32 bits. It may be painful in some cases, but it can be done.

> No
> equivalents to the string instructions.

So, you execute a sequence of instructions in a loop that do the same thing.

> No stack register.

Use a different register for this purpose. Or use a memory location.

Re: Two Click disassembly/reassembly

- > Suddenly it
- > looks like an x86 emulator,

No, it looks like a **compiler** rather than an **interpreter** (emulator).
And that means it will run about 2–10x faster than an emulator.

- > and to be quite honest, it would be easier
- > to write one than your mythical "encoder".

Certainly writing an interpreter is far easier than writing a compiler.
But the interpreter is **much** slower.

- >
- >>
- >>>Otherwise what you'll be writing will no more be assembly than an HLL
- >>>like C++ can be assembly.
- >>
- >>
- >>_Evidently_. Yes. And then?
- &