

Re: Why I stop attacking HLA

Source: <http://coding.derkeiler.com/Archive/Assembler/alt.lang.asm/2005-09/msg00415.html>

- *From:* "randyhyde@xxxxxxxxxxxxxx" <randyhyde@xxxxxxxxxxxxxx>
 - *Date:* 6 Sep 2005 07:29:05 -0700
-

Herbert Kleebauer wrote:

>>
>> Create a set of "addition", "subtraction", and "multiplication" tables.
>> Read the value "n" from the user, and produce an n x n table for each
>> of these problems (using add, sub, and intmul). Based on their
>> C/Java/Pascal/whatever knowledge, the logic is fairly trivial. With
>> about four or five machine instructions, a few variable declarations,
>> and a brief examination of AoA Ch2, they can do this.
>
> Now I understand why it takes weeks to write the first program. You
> have written six lines of text, but it is absolutely unclear what exactly
> you want. Please give an exact specification what the program has
> to do, so I can implement it and see why HLL control structures make
> life easier.

If you don't understand how one might write this little problem, then you're missing the necessary prerequisites. I suggest you take a programming course in C before trying to act like you know what you're talking about around here. :-)

>>
>> Again, you're confusing a discussion of what the flag *are* with a
>> decent knowlege of how someone would use these flags.
>
> Before they can use them, they have to know what they are.

And learning all that takes time. Precisely my point.

>>
>> Well, right off the bat you're assuming they know what a flip-flop is.
>> Big mistake.
>
> If I had assumed that they know what a FlipFlop is, I hadn't explained
> what a FlipFlop is: a FlipFlop is a hardware circuit which can store
> a single bit

Sure. For example:

<http://webster.cs.ucr.edu/AoA/Windows/HTML/DigitalDesigna4.html#6293>

Re: Why I stop attacking HLA

And this usually takes place around three weeks into the quarter. Once they've learned other concepts like what a bit is, numeric representation, and other stuff like that. Again, you keep assuming that they know this stuff all on day one, or that you can throw an arbitrary amount of stuff at them on the first day and they will instantly understand it all. Not going to happen.

>
>> Oh wait! What's a "bit"? And "Why are we talking about
>
> Now you want to tell me, that they have mastered a HLL course and
> don't know what a bit and byte is?

Yes. The whole purpose of the "Machine Organization and Assembly Language Programming" course is to teach them what things like "bits" and "bytes" are. Now you're starting to get the picture. Heck, *anyone* could teach an assembly language programming course in a few days if all the students show up already knowing assembly language. There's no trick to that. But when you've got a fresh slate to work with, and a limited amount of time to fill in that slate, you have to organize your material well in order to maximize what the students learn. And it has to be taught in an appropriate order so that you cover all the prerequisite material first.

>
>> hardware? This is a software class!" Again, given the appropriate
>
> Im not speaking about the hardware (how the FlipFlop is implemented) but
> about programming the hardware (how the FlipFlops are set and reset).

Oh, now we throw in "ports" and "hardware" to the equation. You're assuming they understand these things as well?

>
>
>> prerequisite material, sure, students could understand the flags in a
>> few minutes. However, they don't possess those prerequisites.
>
> Then I have to repeat my question:
>
> There are 4 FlipFlops (a FlipFlop is a hardware circuit which can store
> a single bit) which are set or reset by most of the ALU operations. For
> the add/sub instructions the meaning is:
>
> Z: set if the result is zero, cleared otherwise
> N: set to the most significant bit of the result
> C: set if there is an overflow for unsigned operands, cleared otherwise
> V: set if there is an overflow for signed operands, cleared otherwise

Re: Why I stop attacking HLA

- >
- > Now show me, which of the above can't be understood within two
- > minutes by someone who was able to pass high school.

That would be most students taking an assembly language class for the first time. It doesn't matter how many times you ask the same question over and over again. It doesn't matter what parenthetical explanations you offer like the above, without the prerequisite information the students aren't going to understand what you're talking about.

- >
- >
- >
- >>> Z: set if the result is zero, cleared otherwise
- >>
- >> And of what use is that?
- >
- > ??? If I say, "The car is red", then you answer "And of what use is that?".
- > Facts are facts and are independent of any usability. Before you can
- > use something, you have to know the facts.

If you think that instruction consists of "throwing out a bunch of facts" then you would make a terrible teacher. What can I say?

- >
- >
- >>> N: set to the most significant bit of the result
- >>
- >> What's a "most significant bit?"
- >
- > What do they not understand? The "most significant" or the "bit"?

Both, actually.

- > If it is the "bit", then something must be wrong with the HLL course.

Why? The purpose of the HLL course is to teach beginning programming principles, not teach machine organization. The purpose of the machine organization and assembly course is to teach machine organization and assembly language. Again, teaching an assembly course is made a *whole* lot easier if the students learn all the stuff you're expecting in "the HLL course." Of course, if they did that, then that course would do a terrible job of teaching them the HLL because they would have spent all their time learning machine organization rather than the HLL.

Mind you *some* schools do make students take a basic hardware course prior to the assembly course (Cal Poly Pomona does this, for example). Yep, the assembly course gets a bit farther along when you've got things like bytes and bit and two's complement figured out already. But at most schools, it is the *purpose* of the assembly course to teach

Re: Why I stop attacking HLA

this stuff.

- > If it is the "most significant" then the also don't understand:
- > "The most significant digit of the decimal number 23456 is 2".
- > But then I don't understand why such people are at a university.

If they answered "2" to the question above, then why is not "1" the most significant bit of %1001? See? These are exactly the things that must be taught. They are *not* intuitive. And the last time I checked, few HLLs really cared about things like bits and MSB (at least, at the level taught in the first quarter programming course).

- >
- >
- >> "Why is the most significant bit special?"
- >
- > Why does this matter? Facts are facts.

And of what use are those facts. How do you apply them? Why should anyone bother learning these facts if they aren't useful? Again, try teaching a course sometime. I'd love to see your student evaluations.
:-)

- >
- >> "Why is this bit called the 'N' bit?"
- >
- > Why are you called "Randy"? We have to give it a name and "N" is as good as "X" or "Y" (and as they later will see, even better).

You've hit the nail right on the head when you said "later". That's not the right way to present material. Whenever you say "later" you lose the students' interest. You need to organize your material in an appropriate fashion so that they learn from the basics towards the complex with as few "laters" as possible. Otherwise, you just wind up confusing the students. Doesn't matter whether they are facts or not.

BTW, I think you completely missed the point. It's called the "N flag" because it's the "negative flag" and you've *not* explained what "negative" means with respect to the CPU (e.g., that MSb issue).

- >
- >> What does that have to do with
- >> anything?"
- >
- > What has it to do with assembly programming that you are called "Randy"?

Nothing. But the fact that it is called the N flag (as in negative numbers) has a lot to do with assembly. So you'd better have explained what negative numbers are (and two's complement) so that students can

Re: Why I stop attacking HLA

appreciate the significance of the N flag when you tell them that it is a copy of the H.O. bit of the result. Of course, it's going to take a day or two to go through the two's complement numbering system (and binary numbers; don't forget that this is getting taught in the course, too). A day here, a day there, it all adds up.

>
>
>>> C: set if there is an overflow for unsigned operands, cleared otherwise
>>
>> What is an overflow?
>> What is an "unsigned operation?"
>> What does "set" and "cleared" mean?
>
> They have learned e.g. C and don't know that you can get an overflow
> when you add two integers?

No, they do not.

> The have learned C and don't know the
> difference between "int" and "unsigned int"?

After a 10-week course? No.

> They have learned C
> but don't know what it means to set or clear a token?

They don't know what a "token" is.
And no, they don't particularly know what "set" or "clear" means other than the obvious English connotations.

> But they know
> IF/THEN/ELSE/FOR/WHILE?

Sure. And a few other facilities such as I/O, assignment statements, simple variable declarations, and some simple composite data structures (arrays and, perhaps, structs and strings).

Most of the time spent in that first course is spent learning how to *program*. That is, how to solve problems using statements like IF/WHILE/etc., it is *not* spent learning all about the syntax of the particular HLL the course uses.

And this is the *most* valuable thing they can learn in preparation for an assembly course. For if they don't know how to solve problems on the computer, you've got *much* bigger problems than "they don't know what bits and bytes are."

>

Re: Why I stop attacking HLA

Re: Why I stop attacking HLA

>
>>> V: set if there is an overflow for signed operands, cleared otherwise
>>
>> What is the difference between C and V?
>> What is a "signed operand"?
>> What is an overflow?
>
> They have learned e.g. C and don't know that you can get an overflow
> when you add two integers?

No. They don't.
They learn this kind of stuff in the assembly course.

> The have learned C and don't know the
> difference between "int" and "unsigned int"? They have learned C
> but don't know what it means to set or clear a token? But they know
> IF/THEN/ELSE/FOR/WHILE?

See above.

>
>
>
>> Are you starting to get the picture?
>
> Yes I get the picture. It seems you are unqualified for teaching
> assembly programming.

Whatever you say.
Again, go teach some classes. I'd love to see your student evaluations.

And I'd *really* love to here what your department chair says when you tell him or her that the students in the 10-week HLL prerequisite course need to learn all these things that are normally taught in the assembly course. :-)

>
>
>>> Now show me, which of the above can't be understood within two
>>> minutes by someone who was able to pass high school.
>>
>> Because students don't learn machine organization in high school?
>
> The only thing which wasn't explained is the word "bit". Because they
> already have done a HLL course I think I can assume they already know it.

You assume wrong.
The purpose of the assembly course is exactly to teach this kind of stuff.

> They don't need to know anything about machine organization to

Re: Why I stop attacking HLA

Re: Why I stop attacking HLA

> understand the above.

Whatever you say. Go teach some courses and see how far you get.

>>

>> HaHaHaHa!

>>

>> You're assuming they know what a "flip-flop" is.

>

> No, it was explained above.

HaHaHaHa!

I'd love to see the blank looks on your students' faces when you "explained" flip-flops that way.

>

>> You're assuming they know what "execution" means.

>

> That is not specific to programming. Any child has to execute
> the instructions given by the parents.

Sure. And when they get into a computer programming course, they'll still question what all this means.

>

>

>> You're assuming they know what "position xyz" means.

>

> Then they also don't understand something like: "Go into the
> room with the number 123 and ask there for your next task".

Sure. And they won't draw the analogy to computer memory at all. You will have to spend time in class saying things like "Memory is like a set of rooms with room numbers...blah, blah, blah". A few minutes here, a day there, it all adds up pretty quick. Before too long, it's 4-5 weeks into the quarter before the students are capable of writing their first assembly program.

>

>

>> You're assuming they know what an "instruction" is.

>

> Even a simple solder knows what an "instruction" is.

Not a machine instruction.

>

>

Re: Why I stop attacking HLA

Re: Why I stop attacking HLA

- > > You really need to teach an assembly language course at the university
- > > level sometime (or better yet, at the community college level).
- >
- > I get the feeling, that you better never give an assembly language
- > course at the university anymore.

Go ahead, try it yourself if you're such the expert. And please do share with us your experiences. That would be highly amusing.

Your "teaching techniques" will work *great* for people who already know assembly language programming. But believe me, they won't fly with most students.

Cheers,
Randy Hyde

• *Follow-Ups:*

- ◆ **Re: Why I stop attacking HLA**
◇ *From:* Herbert Kleebauer
- ◆ **Re: Why I stop attacking HLA**
◇ *From:* anonymous

• *References:*

- ◆ **Re: Why I stop attacking HLA**
◇ *From:* anonymous
- ◆ **Re: Why I stop attacking HLA**
◇ *From:* randyhyde@xxxxxxxxxxxxxx
- ◆ **Re: Why I stop attacking HLA**
◇ *From:* anonymous
- ◆ **Re: Why I stop attacking HLA**
◇ *From:* randyhyde@xxxxxxxxxxxxxx
- ◆ **Re: Why I stop attacking HLA**
◇ *From:* Herbert Kleebauer
- ◆ **Re: Why I stop attacking HLA**
◇ *From:* randyhyde@xxxxxxxxxxxxxx
- ◆ **Re: Why I stop attacking HLA**
◇ *From:* Herbert Kleebauer
- ◆ **Re: Why I stop attacking HLA**
◇ *From:* randyhyde@xxxxxxxxxxxxxx
- ◆ **Re: Why I stop attacking HLA**
◇ *From:* Herbert Kleebauer

- Prev by Date: **Re: The Value of a CS Degree**
- Next by Date: **Re: Why I stop attacking HLA**
- Previous by thread: **Re: Why I stop attacking HLA**
- Next by thread: **Re: Why I stop attacking HLA**
- Index(es):
 - ◆ **Date**

Re: Why I stop attacking HLA

◆ *Thread*