

Re: interesting use of NEXT SENTENCE vs. CONTINUE

Source: <http://coding.derkeiler.com/Archive/Cobol/comp.lang.cobol/2005-06/msg00377.html>

- *From:* "Pete Dashwood" <dashwood@xxxxxxxxxxxxxxx>
 - *Date:* Sun, 12 Jun 2005 10:56:46 +1200
-

Chuck,

I have some issues with this (as you might expect :-)) but I'm too busy at the moment to do much about it. I will certainly compile the sample code as soon as I can, and post the results here.

I am amazed that, having compiled it, you didn't do this... They were simple displays.

And why did you convert it to COBOL 74?

I am not arguing about the time that an altered branch takes being different from the the time an unaltered one takes, neither am I in any way suggesting that ALTER is a good construct for use in modern systems.

It is a conceptual thing and I want to know why something that logic tells me SHOULD be so, apparently isn't, on Unisys hardware.

Show me the money. Post the test results. After that we can discuss implications for Unisys.

Pete.

Top Post, more during the coming week...

"Chuck Stevens" <charles.stevens@xxxxxxxxxxx> wrote in message [news:d8coan\\$4e\\$1@xxxxxxxxxxxxxxxxxxxxxxxxxxxx](mailto:news:d8coan$4e$1@xxxxxxxxxxxxxxxxxxxxxxxxxxxx)

>

> "Pete Dashwood" <dashwood@xxxxxxxxxxxxxxx> wrote in message news:3gsc9uFe0rs5U1@xxxxxxxxxxxxxxxxxxxx

>>

>> "Chuck Stevens" <charles.stevens@xxxxxxxxxxx> wrote in message [news:d89the\\$2231\\$1@xxxxxxxxxxxxxxxxxxxxxxxxxxxx](mailto:news:d89the$2231$1@xxxxxxxxxxxxxxxxxxxxxxxxxxxx)

>>>

>>> "Pete Dashwood" <dashwood@xxxxxxxxxxxxxxx> wrote in message news:3gpibiFdiejoU1@xxxxxxxxxxxxxxxxxxxx

>>>>

>>>> I am asking how a machine language BRANCH instruction (which has a

Re: interesting use of NEXT SENTENCE vs. CONTINUE

>> SINGLE
>>> operand), can be SLOWER than a machine language COMPARE instruction
> that
>>> has
>>>> TWO operands.
>>>
>>> We don't really have general "compare" instructions as you seem to
>> describe
>>> them. We have unconditional and conditional branches in both static
and
>>> dynamic forms (BRUN, BRTR, BRFL, DBUN, DBTR, DBFL), and the
conditionals
>> are
>>> based on the low-order bit of the word on the top of the stack,
however
>> that
>>> might get there.
>>
>> I don't want to say ITSA so I'll say ITSLIKE...
>> The Branch on condition (BC) in BAL.
>
> My IBM S/360 PoO has long since faded into the sunset, so I must postulate
> the existence of an instruction to draw the proper comparison.
>
> The BC seems to correspond loosely to our BRTR and BRFL instructions. I
> seem to remember a BR <n> instruction; what I don't remember if there was
a
> BCR (or BRC) instruction, where you might (or might not) branch to the
> memory address pointed to by register <n> based on the tested condition.
>
> This is (or would be) *something like* the dynamic branch used on our
> system, except that the "register" is actually the Top of Stack, and the
> boolean value being tested is underneath it.
>
> While the contents of one of the General Purpose Registers back in S/360
> (and where I encountered it more often, on an Interdata 3) are
> straightforward, the contents of a PCW as used in a DBxx instruction are
> not, they represent considerable indirection, the resolution of which is a
> combined responsibility of hardware and system software. A DBxx
> instruction is *at very best* only somewhat slower than a BRxx
instruction.
>
>
> I converted the program to COBOL74 and ran it using your two examples,
plus
> a third in which an ALTER was *actually used*. The object code for the
> "busywork" incrementing of J is the same.
>
> The case in which a numeric on-stack value for 1 on each update of J
> consumed approximately 1% more time than the case that included a pure
local

Re: interesting use of NEXT SENTENCE vs. CONTINUE

- > branch in the same location.
- >
- > The case in which the first paragraph had been ALTERed to proceed to the
- > second consumed very nearly three times as much processor time as either
- > of
- > the other two.
- >
- > The difference between the "pure local branch" case and the "altered
- > branch"
- > case is precisely the differing object code for the unconditional branch,
- > namely, NAMC, DBUN instead of BRUN.
- >
- > In our environment it is *significantly* more costly to execute a GO TO
- > that's the target of an ALTER than it is to execute a GO TO that is not
- > the
- > target of an ALTER. On a NX6830 it's about THREE TIMES as costly.
- >
- > Even fairly complex conditional expressions can be shown to be less
- > expensive than this. YMMV, but I'm pretty sure it will vary based *only*
- > on
- > the time consumed for evaluation of the conditional, which can be reduced
- > to
- > negligibility through informed coding.
- >
- >>
- >> Your 'low order bit' corresponds to the 'condition code' set in IBM
- >> architectures. Except that you only have two states.
- >>
- >>> There are lots of comparison things that might put a truth
- >>> value on top of the stack, and they're of varying complexity and cost.
- >>
- >> Yes, and they must have at least two operands? Comparison is meaningless
- >> without two things to compare...
- >
- > When the comparison is trivial — get a value, branch on true or false.
- > VALC, ZERO, EQU is also really, really inexpensive, at least on our
- > larger
- > systems.
- >
- >>
- >> And these two things must be in memory. We are not considering register
- > to
- >> register comparisons here, unless we consider unconditional branch on
- >> register address against them. Let's keep oranges and apples in separate
- >> baskets...
- >
- > What's a register? ;-) Everybody knows that a data item is either on
- > the stack or it's not, and all numeric comparisons are done on the stack
- > anyway. "Informed coding" as mentioned above includes ensuring that
- > critical-path data items are declared on the stack in the first place.
- >

Re: interesting use of NEXT SENTENCE vs. CONTINUE

>> This is where I argue that the TIME to do that compare cannot be faster
> than
>> the time to do your branch (irrespective of the branch being
unconditional
>> or conditional, or static, or dynamic.)
>
> Proven wrong in my experience.
>
>> My contention is regarding machine instructions: Namely, comparison and
>> branch.
>
> A dynamic branch -- conditional or otherwise -- can be quite a bit more
> expensive than a simple condition evaluation, depending on the nature of
the
> expression and of its components.
>
>
>> If your 'comparison' breaks down into a series of steps (and you are not
> the
>> only architecture where this happens) then we need to get the cycle time
for
>> each of the steps.
>
> That depends on how "smart" the particular model of machine is in
analyzing
> and executing groups of instructions. The "cycle time" for each step
taken
> in isolation may be utterly meaningless; what counts is the overall
resource
> consumption of the construct, and in the case of a GO TO statement that's
> been ALTERed, it's greater than that of a GO TO that hasn't, but a good
> margin.
>
> The question is whether an *unconditional branch* is always faster than a
> *comparison*, and the answer is "no, not always, and it can be a whole lot
> worse."
>
>> OK, why not get timings for both? Presumably instruction cycle times are
>> documented?
>
> No, not all machines execute their instructions linearly, and instruction
> cycle times are specific only to individual processor models, not to the
> entire architecture.
>
>> It should be easy to look in the documentation and get the cycle times.
>
> That presumes that "cycle times" for instructions taken in isolation are
> meaningful. In your universe they may well be, but ...
>
>> The whole of my contention is that comparison MUST take longer than
>> branching. Nothing to do with ALTER (any more...:-))

Re: interesting use of NEXT SENTENCE vs. CONTINUE

Re: interesting use of NEXT SENTENCE vs. CONTINUE

>
> Then the whole of your contention is **wrong** as a generalization.
>
> My experiments prove that at least in **some** subset of the universe of
> systems, an unconditional branch that happens to be "dynamic" not only
takes
> a whole lot more time than an unconditional branch that happens to be
> "static" but also a whole lot more time than a simple numeric comparison
> taken in combination with a static conditional branch. The difference
is
> significant.
>
> And I agree that it is not merely a matter of ALTER; if the code
associated
> with the label BT001 in your example happens to be in a different code
> segment (NOTE: that's not necessarily the same thing as in a different
> SECTION) the same performance difference will be observed. The point
here
> is that with ALTER, as with a cross-segment branch, the static branch
> **cannot** be used.
>
>> And feel free to rearrange this code (after you have run it as is...:-))
> to
>> cause any of the situations you discussed with page segments, dynamic
>> branching, and so on...
>
> Couple of bugs in it, as written. Won't compile as is ... ;-)
>
> There is no significant difference between the behavior of an ALTERed GO
TO
> statement and a GO TO statement that is determined at compile time to
point
> to a different code segment; the mechanisms are the same, and they are
> equally costly. Most GO TO statements prove to be within the same code
> segments as their destinations, whether ALTERed or not.
>
> -Chuck Stevens
>
>
>

• *Follow-Ups:*

◆ *Re: interesting use of NEXT SENTENCE vs. CONTINUE*

◇ *From:* Chuck Stevens

Re: interesting use of NEXT SENTENCE vs. CONTINUE

• **References:**

- ◆ ***interesting use of NEXT SENTENCE vs. CONTINUE***
 ◇ From: Frank Swarbrick
- ◆ ***Re: interesting use of NEXT SENTENCE vs. CONTINUE***
 ◇ From: Richard
- ◆ ***Re: interesting use of NEXT SENTENCE vs. CONTINUE***
 ◇ From: William M. Klein
- ◆ ***Re: interesting use of NEXT SENTENCE vs. CONTINUE***
 ◇ From: Richard
- ◆ ***Re: interesting use of NEXT SENTENCE vs. CONTINUE***
 ◇ From: docdwarf
- ◆ ***Re: interesting use of NEXT SENTENCE vs. CONTINUE***
 ◇ From: Pete Dashwood
- ◆ ***Re: interesting use of NEXT SENTENCE vs. CONTINUE***
 ◇ From: Howard Brazee
- ◆ ***Re: interesting use of NEXT SENTENCE vs. CONTINUE***
 ◇ From: Pete Dashwood
- ◆ ***Re: interesting use of NEXT SENTENCE vs. CONTINUE***
 ◇ From: Chuck Stevens
- ◆ ***Re: interesting use of NEXT SENTENCE vs. CONTINUE***
 ◇ From: Pete Dashwood
- ◆ ***Re: interesting use of NEXT SENTENCE vs. CONTINUE***
 ◇ From: Chuck Stevens
- ◆ ***Re: interesting use of NEXT SENTENCE vs. CONTINUE***
 ◇ From: Pete Dashwood
- ◆ ***Re: interesting use of NEXT SENTENCE vs. CONTINUE***
 ◇ From: Chuck Stevens
- ◆ ***Re: interesting use of NEXT SENTENCE vs. CONTINUE***
 ◇ From: Pete Dashwood
- ◆ ***Re: interesting use of NEXT SENTENCE vs. CONTINUE***
 ◇ From: Chuck Stevens
- ◆ ***Re: interesting use of NEXT SENTENCE vs. CONTINUE***
 ◇ From: Pete Dashwood
- ◆ ***Re: interesting use of NEXT SENTENCE vs. CONTINUE***
 ◇ From: Chuck Stevens

- Prev by Date: ***Re: Numeric validation revisited***
- Next by Date: ***Re: interesting use of NEXT SENTENCE vs. CONTINUE***
- Previous by thread: ***Re: interesting use of NEXT SENTENCE vs. CONTINUE***
- Next by thread: ***Re: interesting use of NEXT SENTENCE vs. CONTINUE***
- Index(es):
 - ◆ ***Date***
 - ◆ ***Thread***