

Re: Calling a function by hand

Source: <http://coding.derkeiler.com/Archive/Delphi/borland.public.delphi.language.basm/2004-12/0033.html>

From: Avatar Zondertau (avatarzondertau_at_hotmail.com)

Date: 12/08/04

Date: 8 Dec 2004 09:22:30 -0800

> *I need to call a function at runtime but the number of arguments and
> their types, and the pointer to the function are not known until
> runtime.*
>
> *So, I need to call a function and pass the parameters "by hand"
> rather than the compiler doing it at compile time.*
>
> *I was going to create a procedure that accepts the function pointer
> and a variant open array, like the function `Format(", []` does.
> Then I was going to call the function and pass the parameters.*
>
> *I know there are different calling conventions, `stdcall`, `cdecl`,
> etc... I assume that I will be able to call the function correctly
> if I know its calling convention and its parameters. The variant
> array will tell me what the parameter types are.*
>
> *The end resulting procedure I imagine is
> procedure `CallFunction(AFunction: Pointer; Args: array of const);`
>
> *I want this to be able to call functions in C, C++, and Delphi. I
> may also want to call an object method in Delphi or C++.*
>
> *The possibility of getting return results, and var params might be
> necessary.*
>
>
> *What advice can you give me?**

Not to make it more complicated than it already is by using a variant open array.

I would suggest using an open array of pointers instead. Put the parameters in the array by:

- Adding a pointer to it (by reference parameters)
- Typecasting to a pointer, adding the result (32-bit parameters)

borland.public.delphi.language.basm: Re: Calling a function by hand

– Zero extend/sign extend (depending on type) to 32–bits, then typecast to pointer, add the result (8–bit and 16–bit parameters)

– Split in multiple 32–bit blocks, typecast those to pointers, then add those (more than 32–bit parameters)

This way you can just put the array on the stack and you don't have to care about the parameter types anymore. If you really want to use a variant open array–supporting function, just make a wrapper to the array of pointer function. This can be done in Pascal.

I also suggest you only use stdcall, ince it's widely supported and probably the easiest calling convention.

Return values are stored in AL, AX, EAX or EDX:EAX (depending on result size). If you don't touch EAX and EDX after calling the function you can simply ignore them and return an Int64. Cast this return value to the proper type after making the call.

Keep in mind that return values can sometimes be passed as an additional by reference parameter. This is for example the case when 64–bit+ structures are returned.

Calling methods just means adding an extra parameter – Self – to the parameter list.