

Re: ml64, PROC and parameters

Source:

<http://www.tech-archive.net/Archive/Development/microsoft.public.development.device.drivers/2008-01/msg00052.html>

- *From:* "Ivan Brugiolo [MSFT]" <ivanbrug@xxxxxxxxxxxxxxxxxxxxxxxx>
 - *Date:* Wed, 2 Jan 2008 13:48:31 -0800
-

#1

_test_nested creates 98h bytes of stack space for the sake of the exercise. It does not use them, but, the stack space could be used for temporary storage, temporary structures, etc etc

#2

exception handling support in amd64/ia64 is not optional. Every non-leaf function needs to have a function descriptor (or function-entry, as sometimes referred in older documentation, and in the `.fnentry`` command in `cdb/ntsd/windbg`, or `personality-routine`, if you have ever read the ABI for IA64 from HP/Intel). The function descriptor is used to perform stack unwinding even if your function does not throw/raise/handle an exception. Imagine the following two scenarios:
-stack probing and commitment
Your function might take an exception while pushing the registers on the stack while in the middle of the prologue (because it hits the guard-page)
-nested calls
Your function may call one other function that raises an exception, and, your function is being called from a function that handles the exception.
In this case the compiler needs to generate a function-entry header in order for the Exception Unwind code to locate the filter.

#3

I simply wanted to make your function to assemble as it were. Indeed `_MyProc` is wrong by the rules and principles set so far. You should push on the stack only the non-parameter register.

--

--

This posting is provided "AS IS" with no warranties, and confers no rights. Use of any included script samples are subject to the terms specified at <http://www.microsoft.com/info/copyright.htm>

Re: ml64, PROC and parameters

"Peter" <Peter@xxxxxxxxxxxxxxxxxxxxxxxxxxxx> wrote in message
news:55EDEBA5-4AFB-425A-83E2-7BD8F69B7B5F@xxxxxxxxxxxxxxxxxxxx

Thanks for more lite on this problem.

I think I understand what is register homing now, and that first four parameters are stored in mentioned four registers (which is new for me).

Q1:

In `_test_nested` function, Why is there created 98h bytes space on the stack ?

What purpose can be used this stack space for ?

Q2:

keyword "frame": I red paragraph about it in ml64. If I good understand it has sense to use it only when I want support for structured exception handling ? Or it has also other sense ?

Q3:

I am a bit confusing with your sentence: "You do NOT save the parameter registers by pushing them on the stack..." and with prologue in `_MyProc`, because there are pushed two registers: `rcx, rdx` and these two can be also used for parameters. If `_MyProc` is changed to have parameters, can stay prolog in `_MyProc` sample the same ?

Peter

"Ivan Brugiolo [MSFT]" wrote:

First of all, I would advise you using the macros already defined in `macamd64.inc`.
Then, I wrote the code you posted in a way that it assembles correctly, and I reported it below.
Notheless, what you are trying to do just tells me you have not quite understood the calling convention of the architecture.
The first 4 parameters are in registers `rcx, rdx, r8, r9`.
You do NOT save the parameter registers by pushing them on the stack. The caller has prepared space on the stack for you, and, all you have to do is register homing.
The `_test_leaf` function shows how to do register homing properly.
In any case, register homing is hardly ever necessary, and, it usually indicates poor optimization and register handling.
You can push on the stack the other registers besides the first 4, and, that is up to your function to do that via an explicit ``push rbx`` call, or something more complicated.
The fifth param of a function is usually at ``rsp+28``, unless you have altered the stack pointer in the prologue.

//-----

Re: ml64, PROC and parameters

```
include macamd64.inc

LEAF_ENTRY _test_leaf, _TEXT$00

mov [rsp+ 8h],rcx
mov [rsp+10h],rdx
mov [rsp+18h],r8
mov [rsp+20h],r9

ret
LEAF_END _test_leaf, _TEXT$00

public _test_nested
NESTED_ENTRY _test_nested, _TEXT$00

sub rsp, 98h
.allocstack 98h
END_PROLOGUE

mov rcx, 1
mov rdx, 2
mov r8, 3
mov r9, 4

call _test_leaf

add rsp, 98h
ret
NESTED_END _test_nested, _TEXT$00

_TEXT$00 segment para 'CODE'
public _MyProc
_MyProc proc frame
push rbx
.pushreg rbx
push rcx
.pushreg rcx
push rdx
.pushreg rdx
sub rsp, 98h
.allocstack 98h
.endprolog

mov rcx, 1
mov rdx, 2
mov r8, 3
mov r9, 4

call _test_leaf

add rsp, 98h
```

Re: ml64, PROC and parameters

```
pop rdx
pop rcx
pop rbx
ret
_MyProc endp
_TEXT$00 ends
```

END

--

--

This posting is provided "AS IS" with no warranties, and confers no rights.

Use of any included script samples are subject to the terms specified at <http://www.microsoft.com/info/cpyright.htm>

"Peter" <Peter@xxxxxxxxxxxxxxxxxxxxxxxxxxxx> wrote in message
<news:C935B3A1-9216-4014-9EF6-3E1096FB0659@xxxxxxxxxxxxxxxxxxxx>

Thanks Ivan,
I tried to add prologue to my function, but still I have
problem to
compile
simple example function, it complains: "A2008 syntax error :
FRAME".
I am looking into MSDN MASM Reference -> PROC.
There is "FRAME" after
arguments in PROC definition, I dont understand what is
wrong with my
asm
sample.
I looked for some more detailed x64 MASM doc, but
nothing found, in
MSDN
are
only several examples, not helpful for this case.
Can you recommend some links/doc for x64 MASM (I found
only something
for
x86) ?
This is my not-compileable sample, what is wrong with it ?
////////// x64 example
.code

```
MyFunc PROC dwArg0:DWORD, dwArg1:DWORD
FRAME
push rbx
.pushreg rbx
```

Re: ml64, PROC and parameters

```
push rcx
.pushreg rcx
push rdx
.pushreg rdx
.endprolog

; loading first argument:
mov eax, dwArg0
;... another work

functionend:
pop rdx
pop rcx
pop rbx

ret
MyFunc ENDP
```

```
END
////////// end of ml64 example
Peter
```

"Ivan Brugiolo [MSFT]" wrote:

Can you post a relevant fragment ?

on x64, reading the first 4 parameters from any on-the-stack location without having done the register-homing first leads to undefined results. Starting touching the stack without having completed the prologue of a function gives also undefined results.

On average, you always read the parameters trough a register (for the first 4) or trough a direct stack-read (either via ESP or some other register) only after you have completed the prologue of the function.

--

--

This posting is provided "AS IS" with no warranties, and confers no

Re: ml64, PROC and parameters

rights.

Use of any included script samples are
subject to the terms specified

at

<http://www.microsoft.com/info/cpyright.htm>

"Peter"

<Peter@xxxxxxxxxxxxxxxxxxxxxxxxxxxx>

wrote in message

news:A394335C-4DB8-4C62-9921-E5FE02B2A663@xxxxxxxxxxxxxxxxxxxx

I have created function in
*.asm module included in
x64 project.

Problem is that arguments
cannot be used when they
are defined after

PROC,

for example:

MyFunc PROC

arg1:DWORD,

arg2:DWORD

In such function I cannot
read argument value by:

1. mov ebx, arg1

But no problem to read

argument value by:

2. mov rbx, [rsp+8*1]

I found some information
about this topic in forums
that in ml64

does

not

support passing parameters

by 1. Is it true ? Or I is

something

wrong

with

defining my function ?

Peter

Re: ml64, PROC and parameters