您的潜力，我们的动力

Microsoft
微软（中国）有限公司

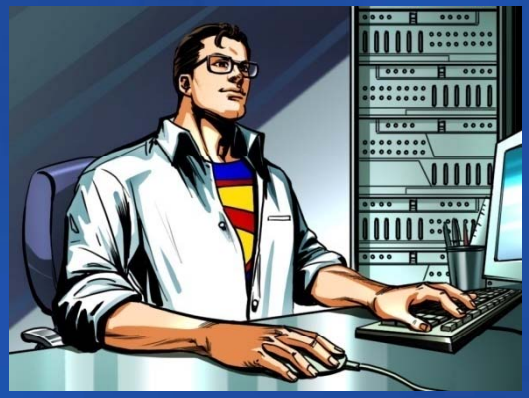
Windows Embedded从入门到精通系列课程

Windows Embedded CE
Windows Mobile
调试与性能优化

Jianchao Teng
Consultant
Microsoft Consulting Services

msdn

MSDN Webcasts



微软中文技术论坛

您的潜力，我们的动力

Microsoft 微软（中国）有限公司

——精彩生活每一天

http://forums.microsoft.com/china

本周活动更新：

- ✓ **Top 10 论坛英雄！**
- ✓ 畅谈我的2007

与众不同：

- ✓ **版主：50+ 微软最有价值专家（MVP）**
- ✓ **涵盖微软几乎所有产品线和知识库**
- ✓ **30+ 适合开发人员和 IT 专业人员技术板块**

City Event Sponsor Program

相聚同城技术培训

微软.NET俱乐部 .NET User Group

MVP Microsoft Most Valuable Professional

msdn

MSDN Webcasts

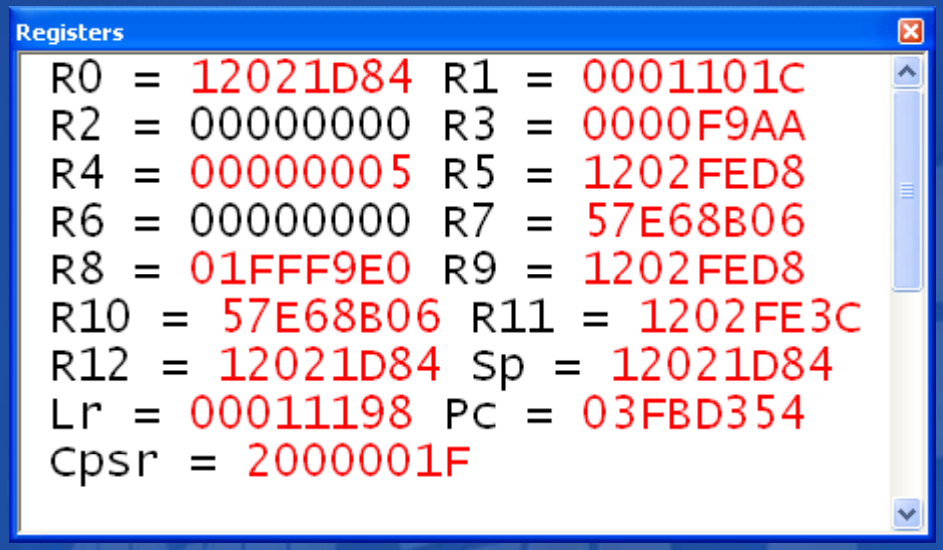
本次课程内容包括

- CE and Mobile Architecture Overview
- Debugging Tips
- Performance & Profiler

CE and Mobile Overview

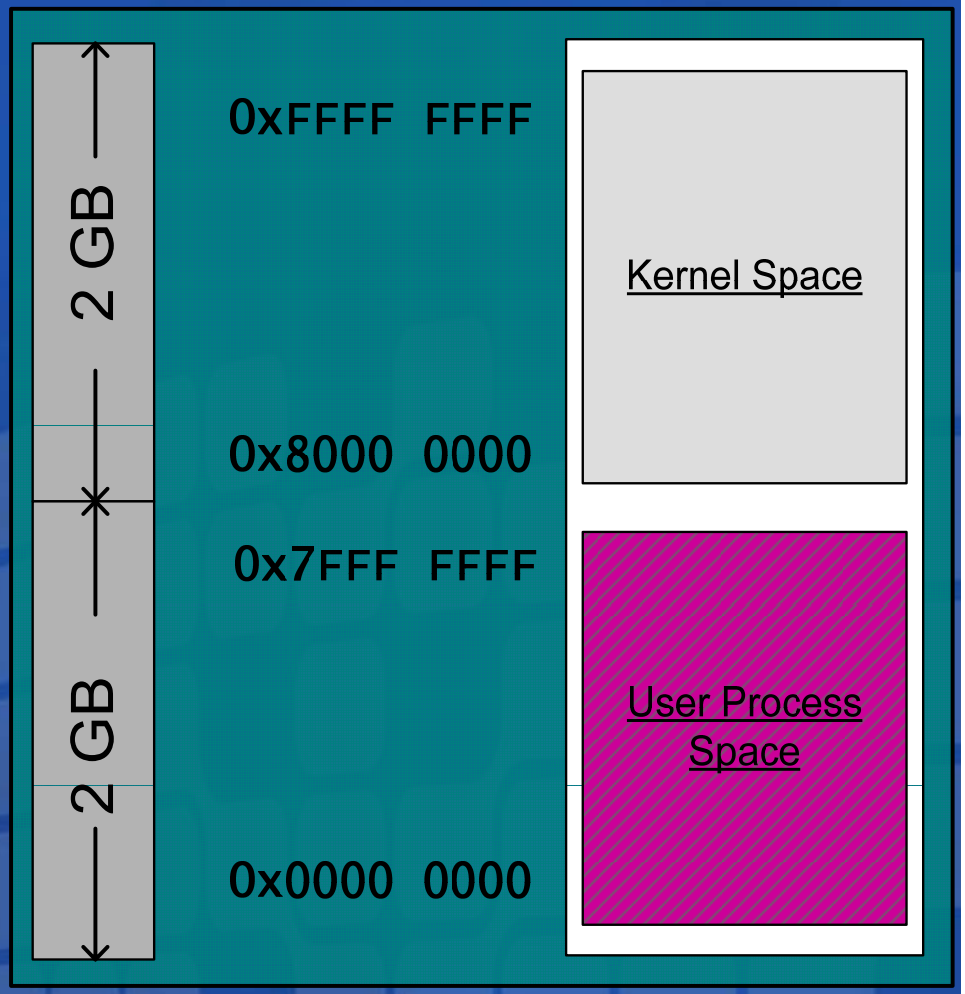
Memory Layout

Hint: Register Window



Windows CE Virtual Memory Review

- 32-bit OS – 4 GB possible
- Top half used by Kernel
- Bottom half used by User

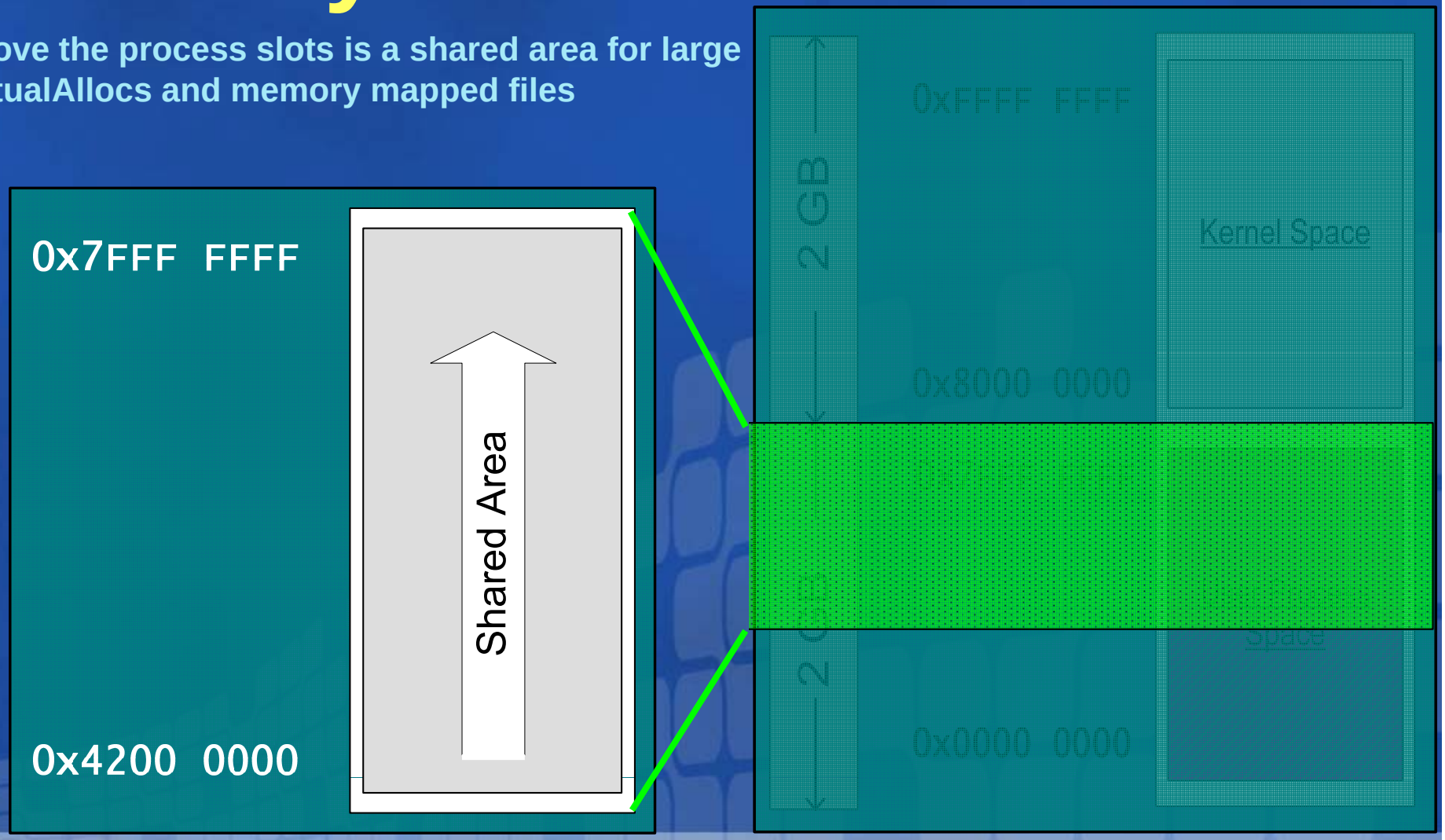


Windows CE Virtual Memory Review

您的潜力，我们的动力

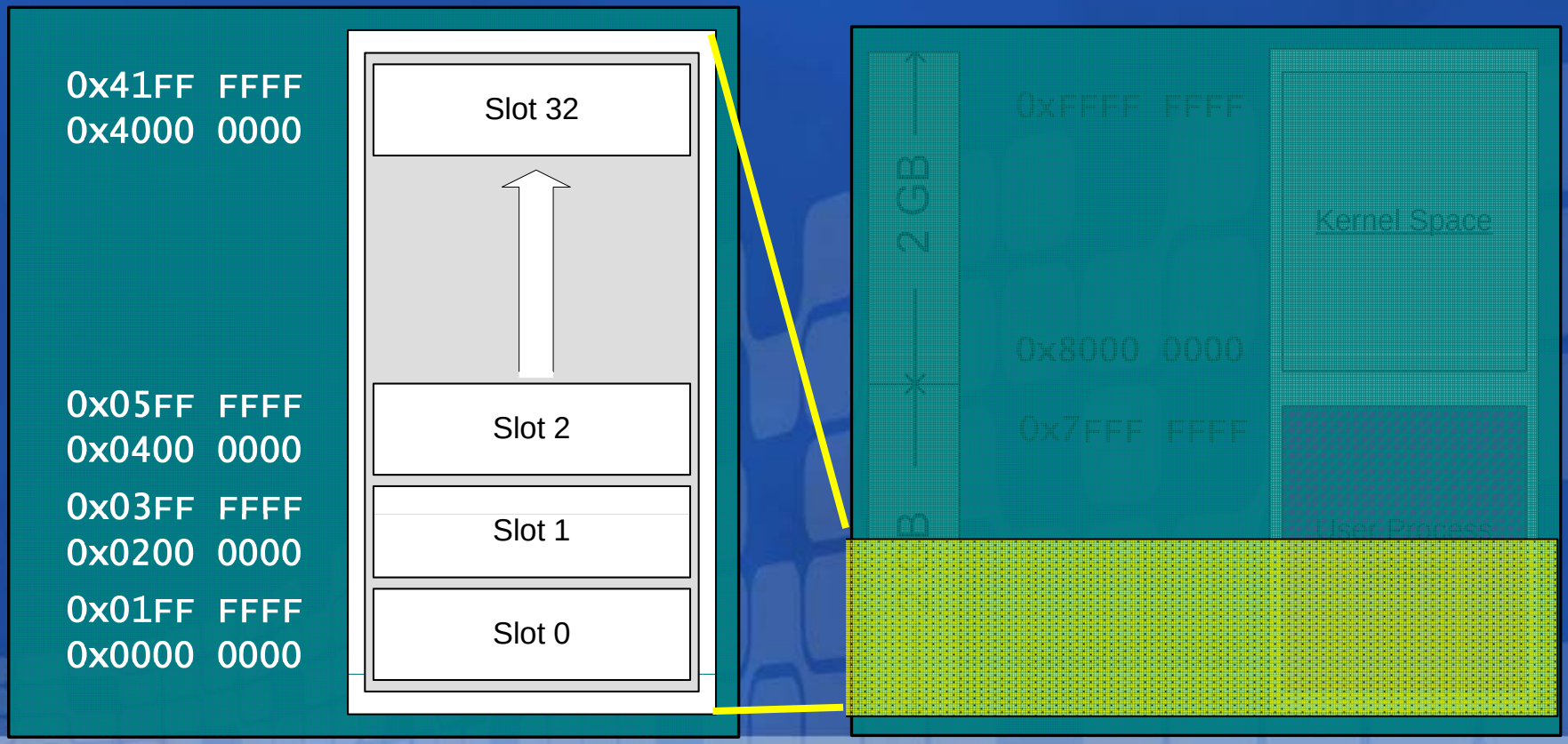
Microsoft
微软（中国）有限公司

Above the process slots is a shared area for large VirtualAllocs and memory mapped files



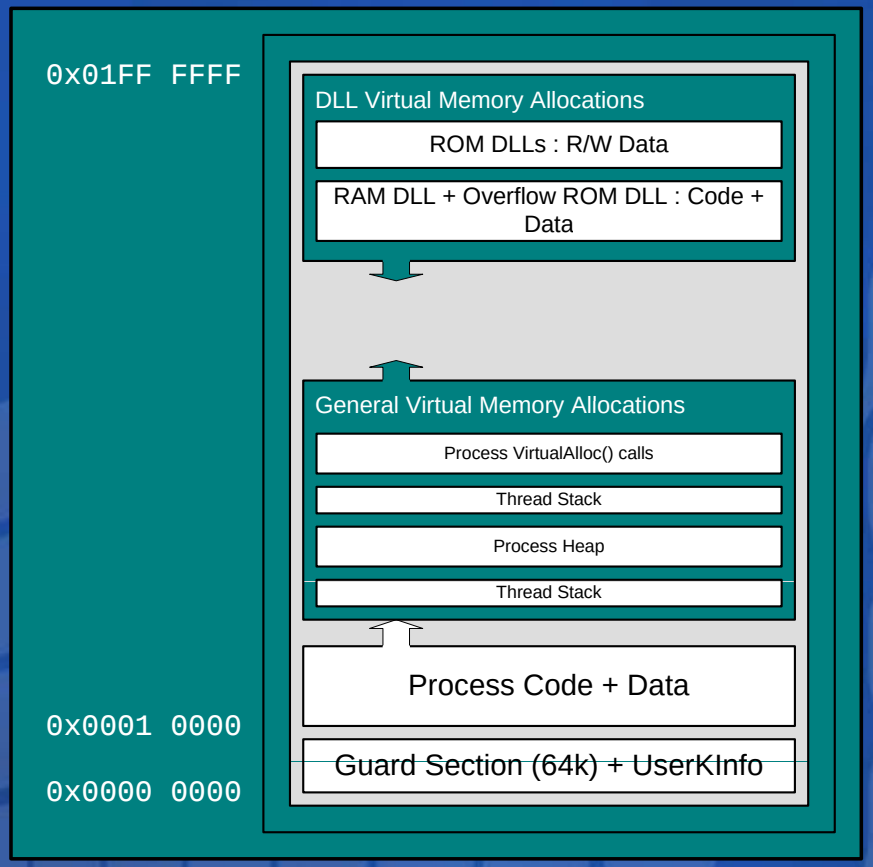
Windows CE Virtual Memory Review

32 MB Process slots take the lowest addresses



Process Slot

- 32MB Process slot is shared by DLL, process, and all of its virtual allocations
- All virtual allocations are 64kB-aligned
- Pages may be committed within a virtual allocation on a page granularity (4kB)
- DLL allocations in a slot start at high addresses and grow down
- Process and general allocations start at low addresses and grow up

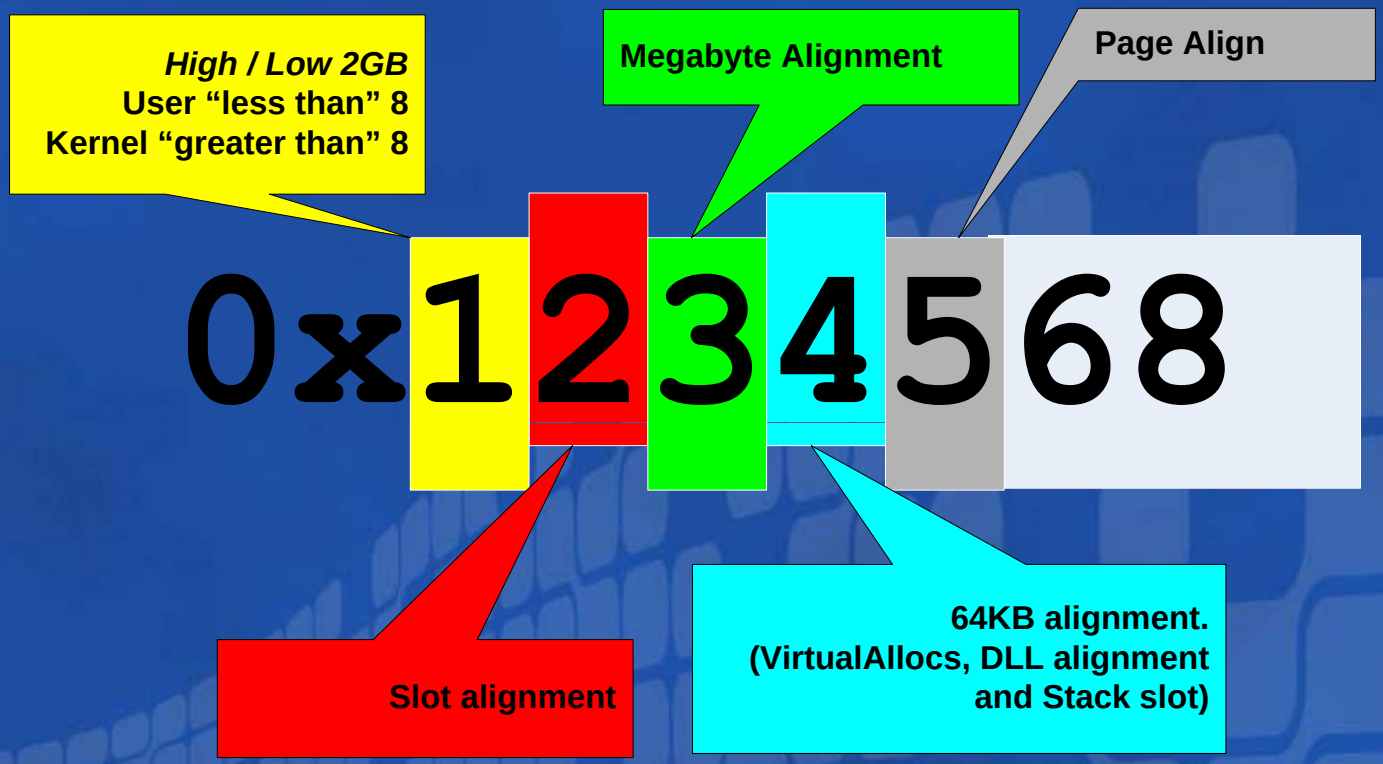


Debugging tips

What do we learn from the memory layout?

Hex Address Anatomy

Hex addresses have identifiable positions:



Hex Address Anatomy

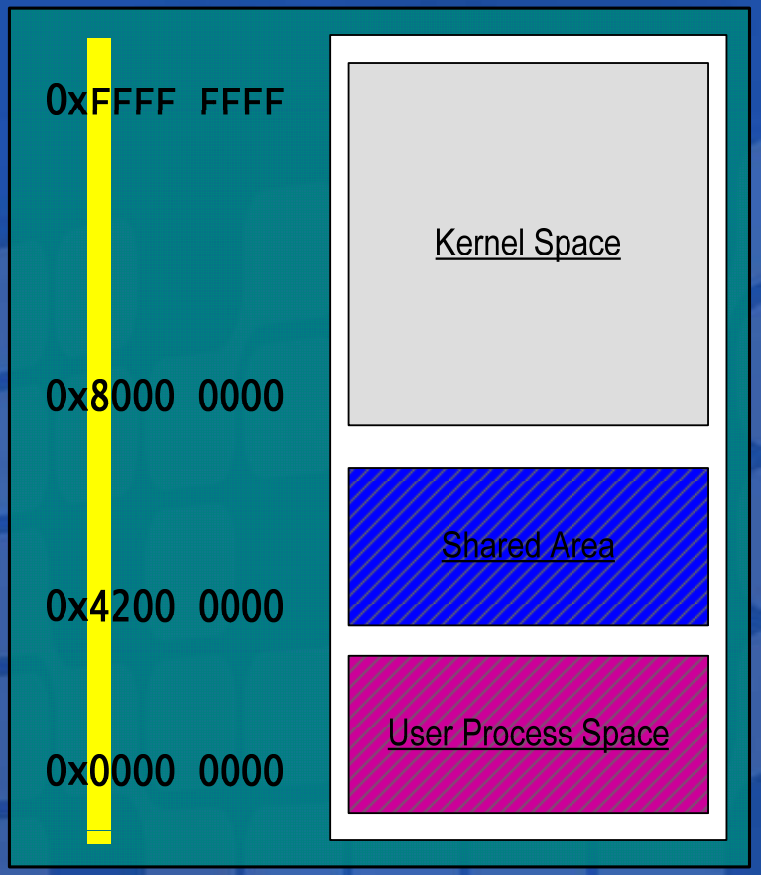
Position 1 (in yellow) tells you if its kernel space or user space

8 and above is kernel

Below 8 is User space

High / Low 2GB
User “less than” 8
Kernel “greater than” 8

0x12345678



Hex Address Anatomy

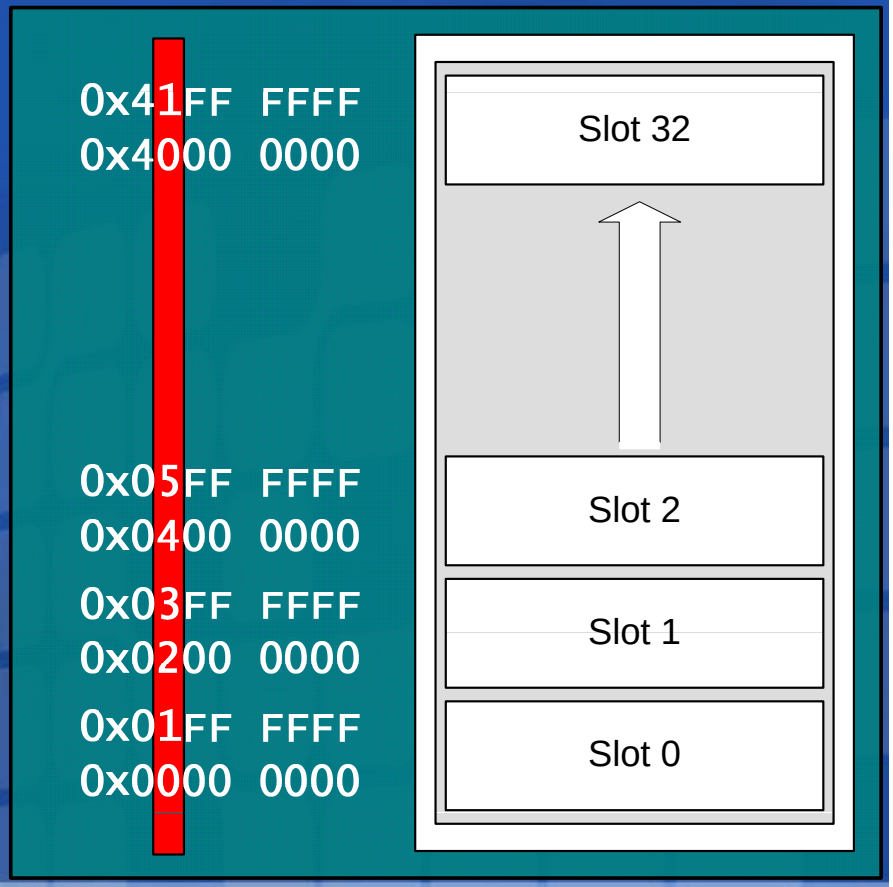
Position 2 (in red) is process slot position

If slot number is odd, address is likely a DLL

If slot number is even, address is likely an app

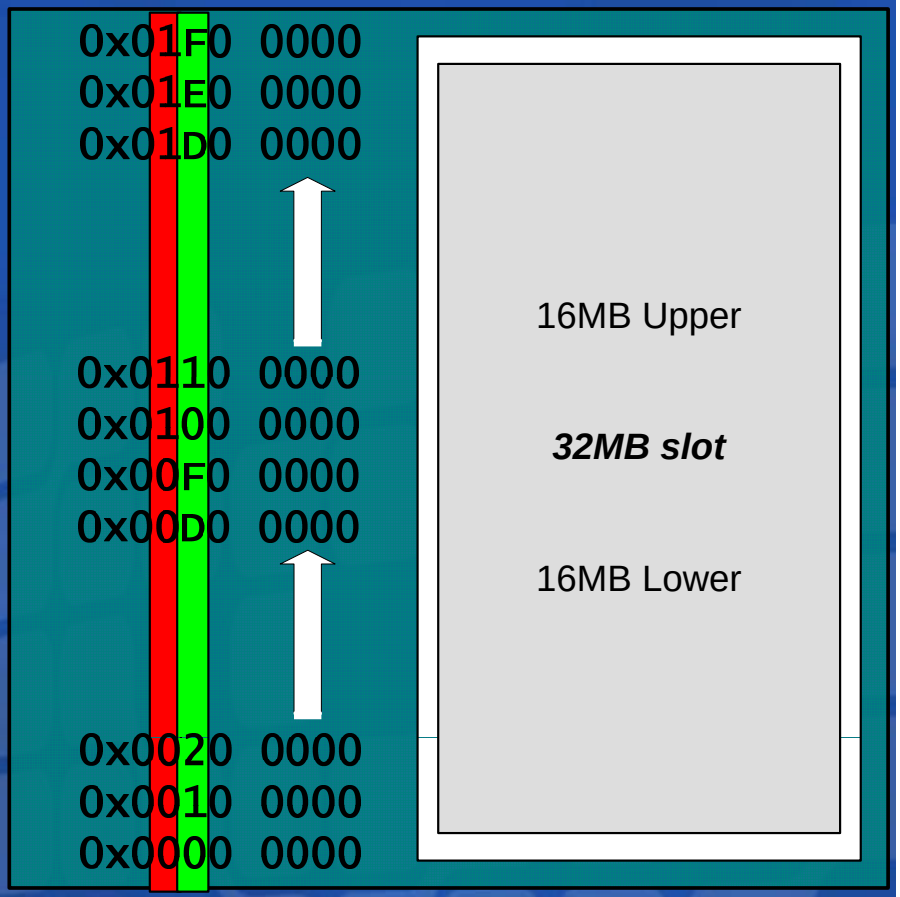
0x12345678

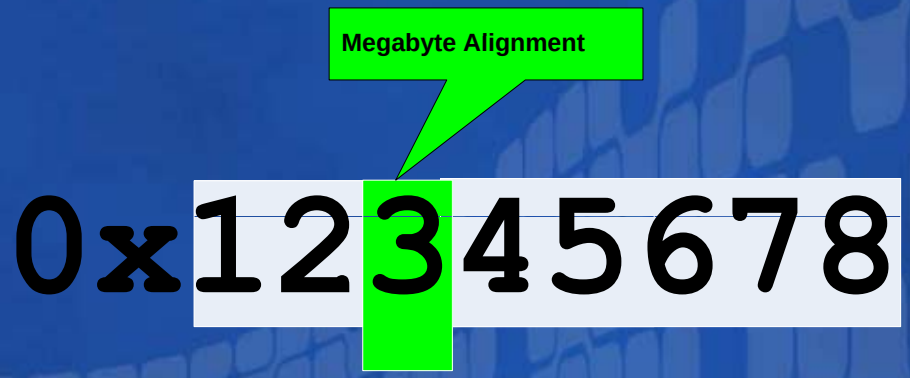
Process Slot alignment



Hex Address Anatomy

32MB in hex is: 0x0200 0000





Hex Address Anatomy

您的潜力，我们的动力

Microsoft 微软（中国）有限公司

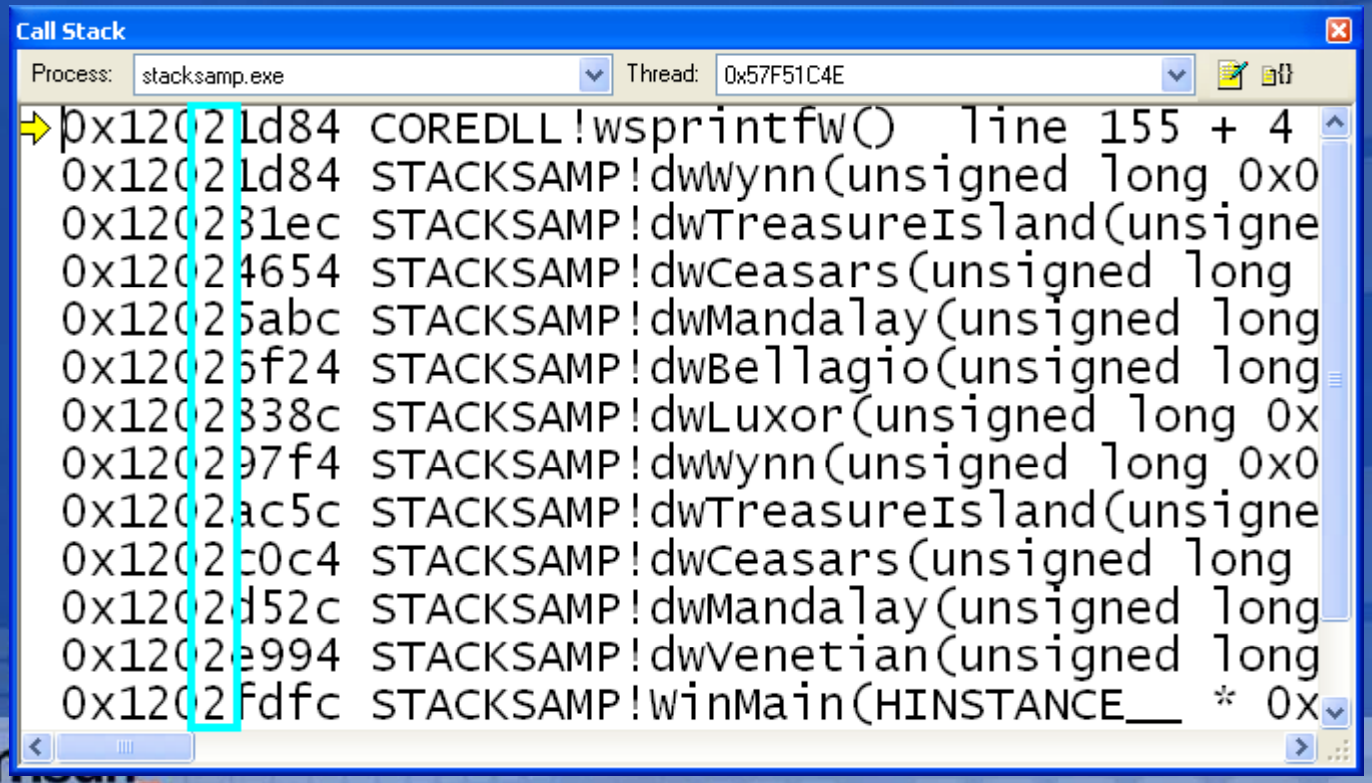
Position 4 (in cyan) is 64 kb alignment

DLL alignment

Stack alignment

0x12345678

64KB alignment. (VirtualAllocs, DLL alignment and Stack slot)



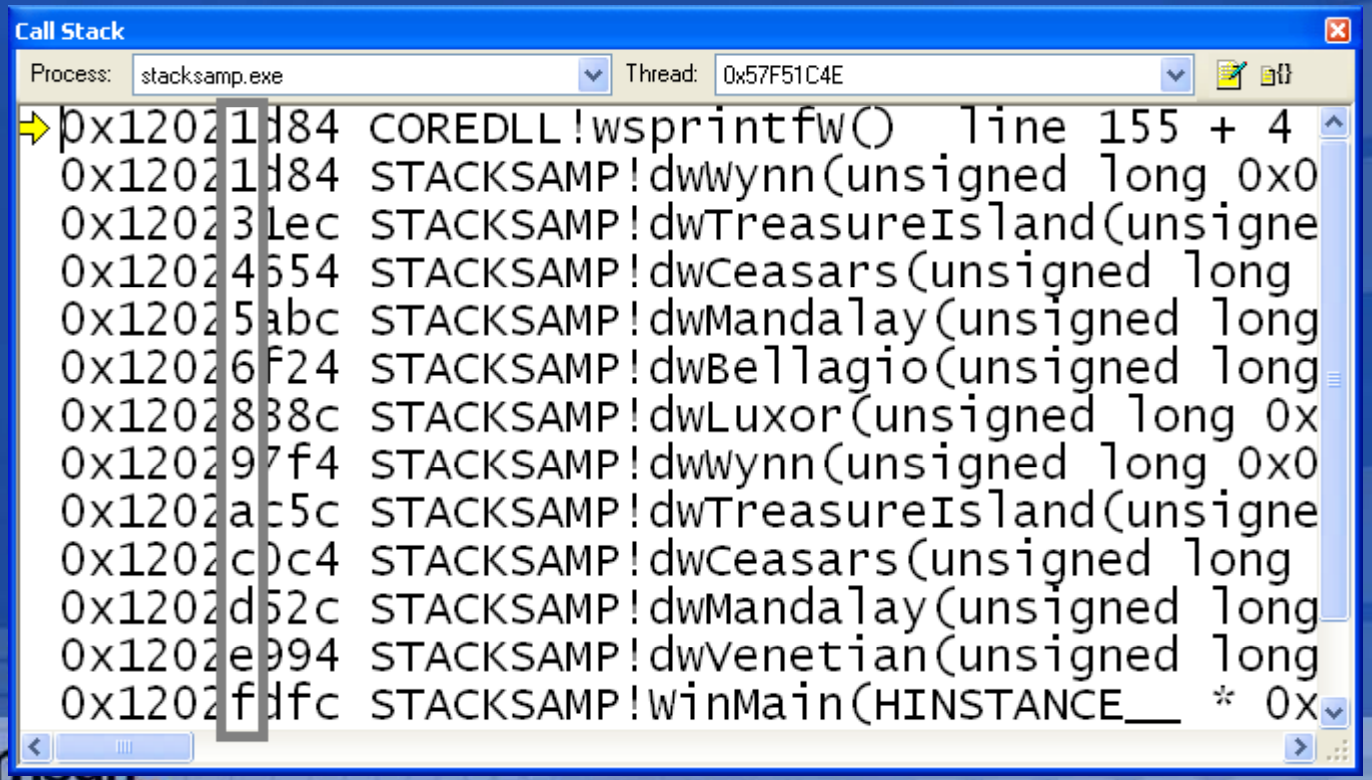
Hex Address Anatomy

Position 5 (in gray) is page (4kb) alignment

16 pages == 64kb

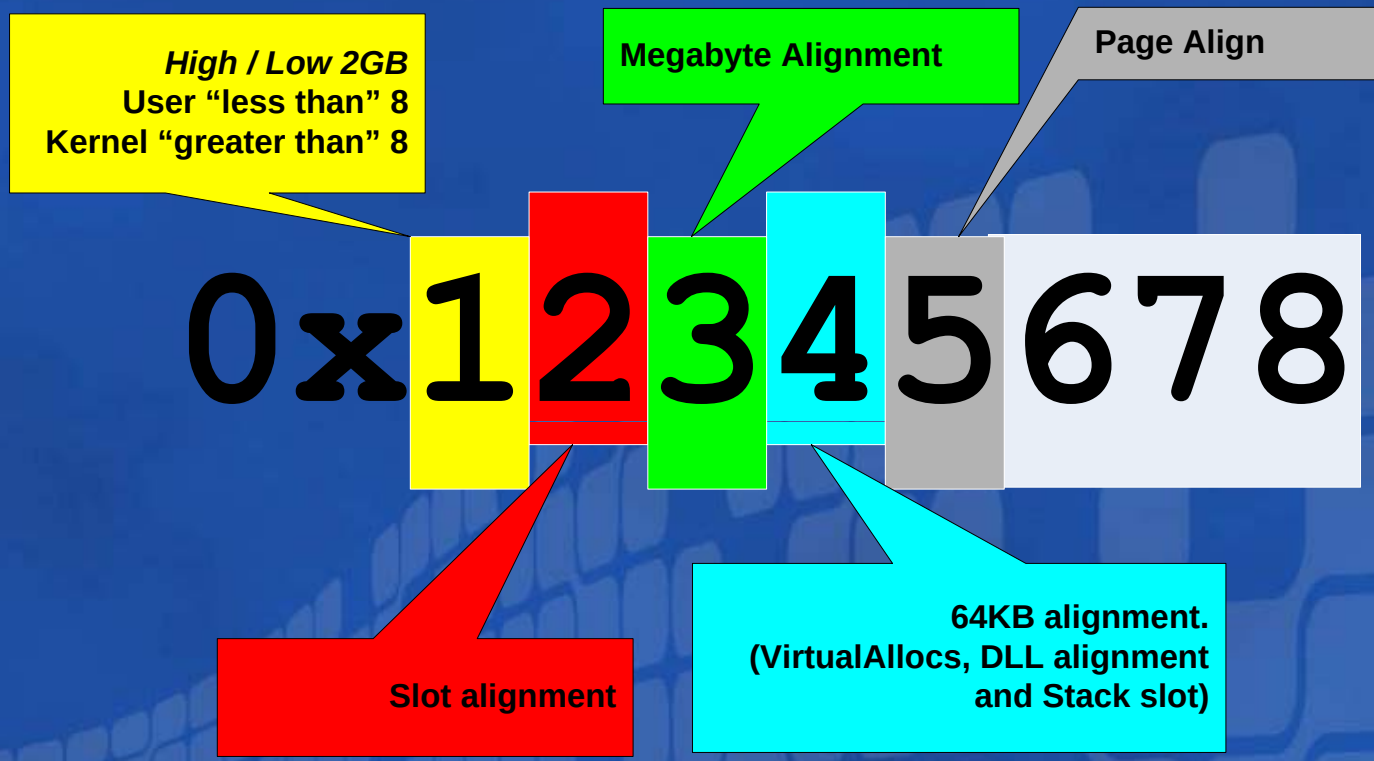
Page Align

0x12345678



Hex Address Anatomy

Lets take another look



Thinking in Hex Quiz 1

- What is the difference between:

```
0x00119235
```

and

```
0x0A119235
```
??

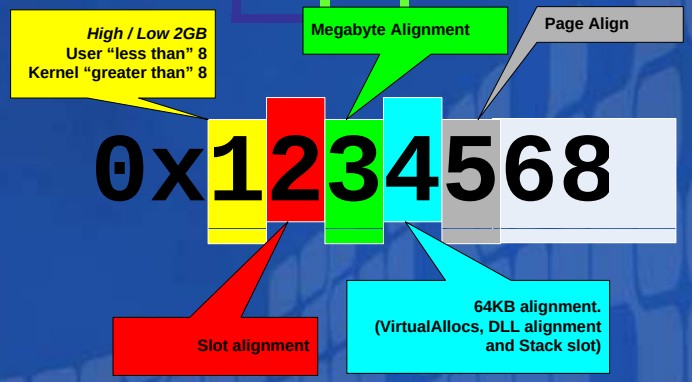
- Extra credit: What do you think it points to?

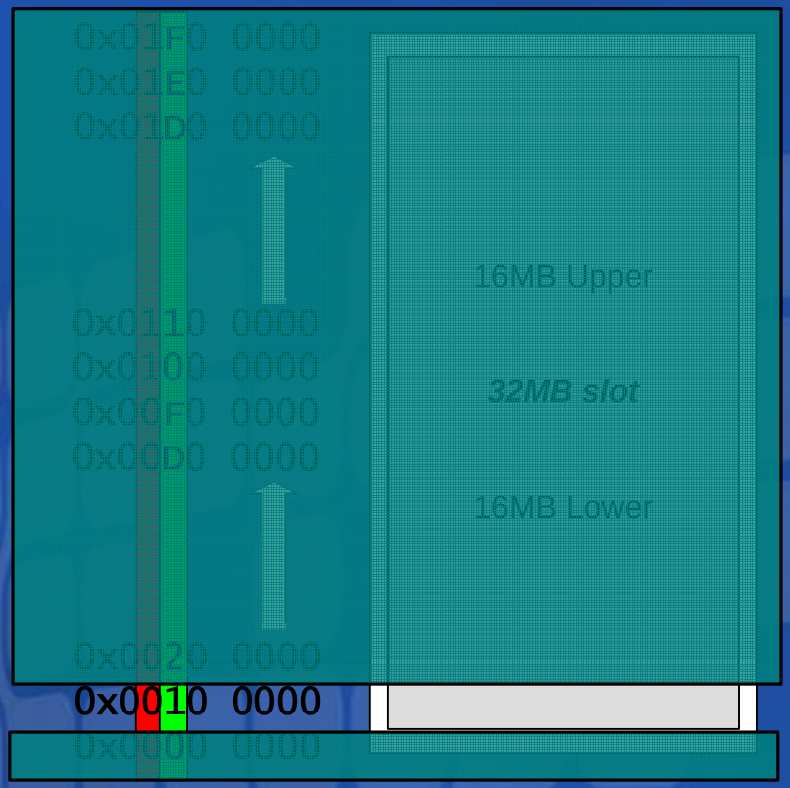
Thinking in Hex (Answer 1)

Same address!

Second it is being viewed outside its process slot.

```
0x00119235
0x0A119235
```





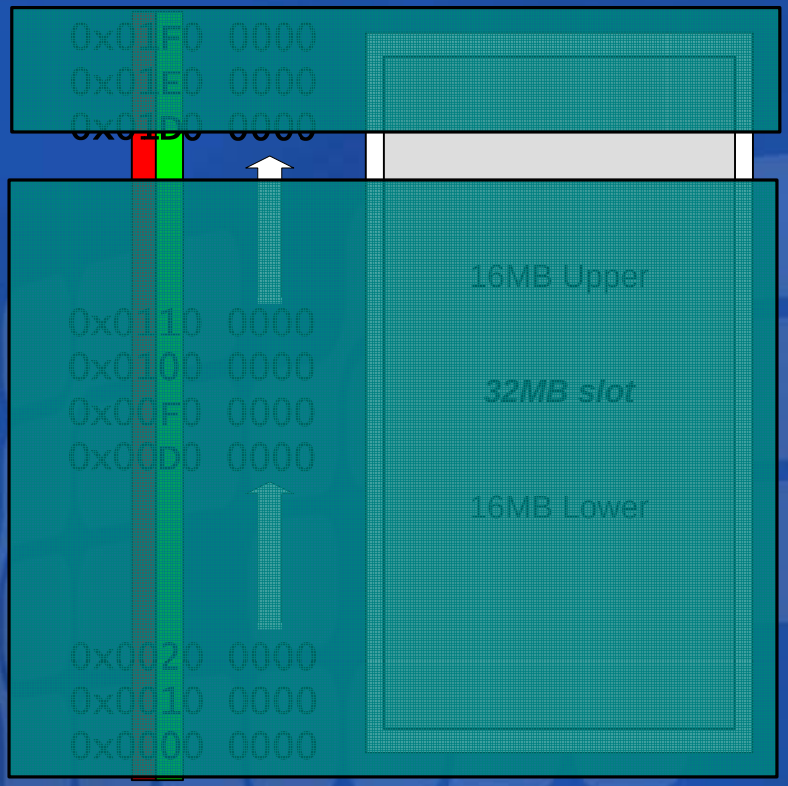
“Most Likely” code or global

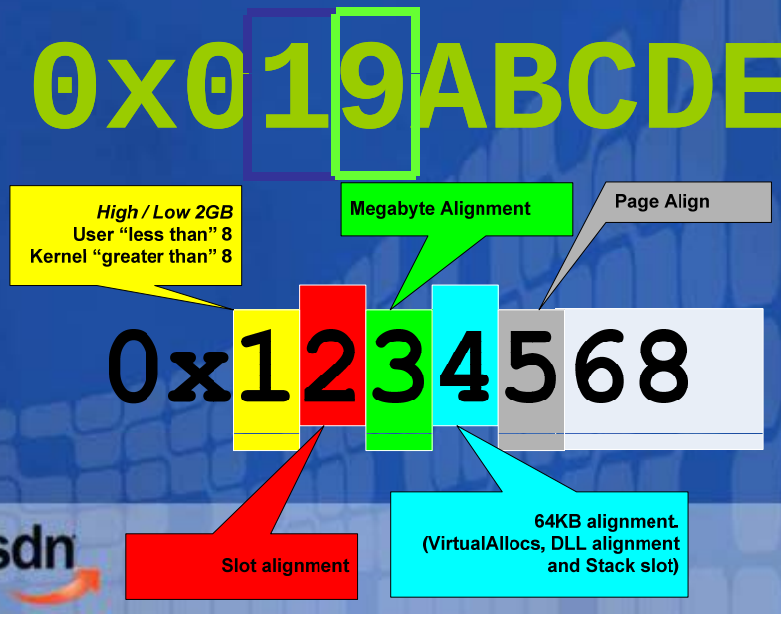
Thinking in Hex (Quiz 2)

- What is: `0x019ABCDE`?

Thinking in Hex (Answer 2)

0x019ABCDE is “most likely” CODE from a DLL.





Visually Debugging Call Stacks

Stacks grow from higher to lower addresses

Default size of ARM stack is 64KB and they are always 64KB aligned. This means that the “top functions” of a stack will be located very close to a 64 KB boundary.

Windows CE also employs the notion of two separate 4KB guard pages that throw exceptions if entered

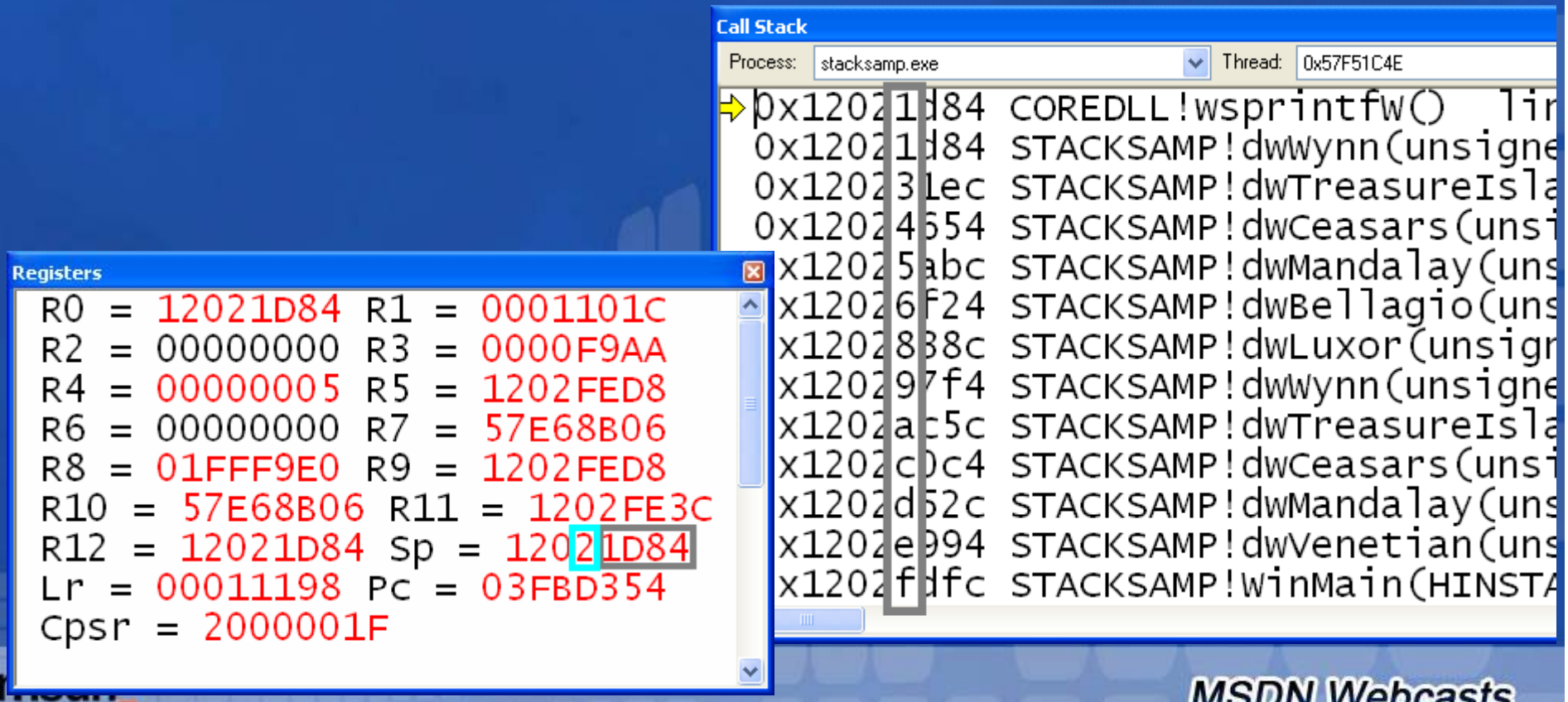
Visual Stack Inspection #1

您的潜力，我们的动力

Microsoft 微软(中国)有限公司

Stack Pointer “dipped” into first guard page

Stack overflow



Under standing Exception

```
: Exception 000 Thread=83e1fc58 Proc=23cd516a 'Div0.exe'
: AKY=00000081 PC=0001109f(Div0.exe+0x0000109f) ESP=1002fbe4 EA=00000000
```

AKY → "Access Key“

PC → "Program Counter”

ESP /SP → “Stack Pointer”

ESP - EA - RA - FSR are CPU specified register or MMU register

Back to address topic:

0x0001109f

Data or code failed!

Using shell.exe to collect runtime Information

- gi command

  gi ["proc","thrd","mod","all"]* ["<pattern>"] : Get Information

  - proc -> Lists all processes in the system
  - thrd -> Lists all processes with their threads
  - delta -> Lists only threads that have changes in CPU times

DEMO

Target Control Debugging

Profiler and Optimization

Optimize what?

How could we collect the runtime information ?

Performance Optimization

Before optimize the system ,We need to know how does the system running

What is bottle-neck of the system?

Which module cost most CPU time?

Which event cause system blocking ？

Kernel Profiler

- What is the profiler mean?
- Data Collection
- Time Based
  - Monte Carlo (time-based sampling)
- Call Based
  - System Calls (call-based)

Monte Carlo

- High resolution timer as the “Hit”
- Interrupts the OS periodically and samples the PC
- void ProfilerHit(DWORD ra);

System Calls

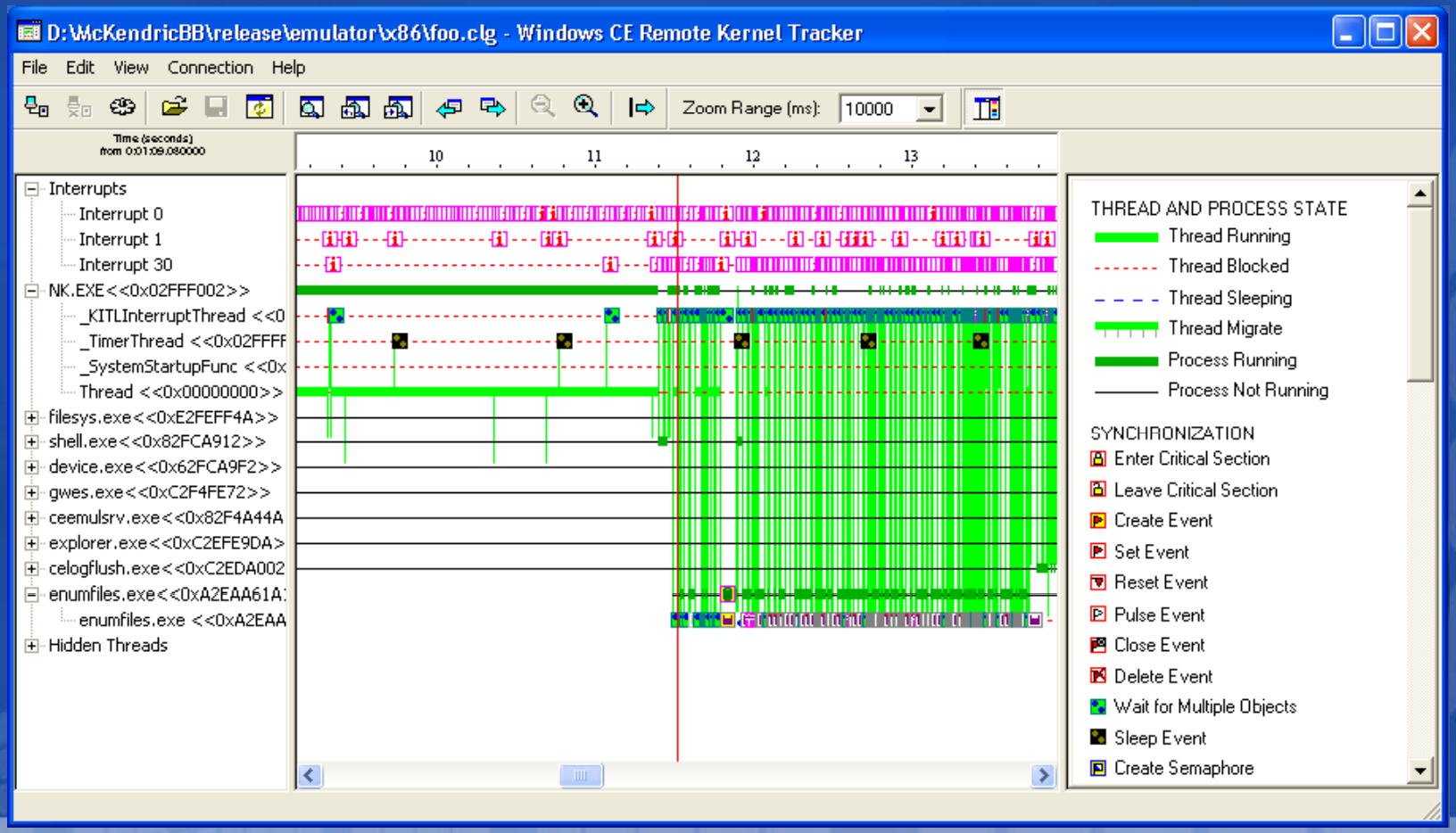
- hits in the ObjectCall function in the kernel
- System call profiling monitors this ObjectCall function

Remote Kernel tracker

- Capture OS events as they happen
  - Thread and process creation/execution/PRIO
  - Synchronization objects
  - Interrupts
  - Memory, paging, TLB misses
  - Loader events
  - Boot events

Examining Thread Behavior

Remote Kernel tracker



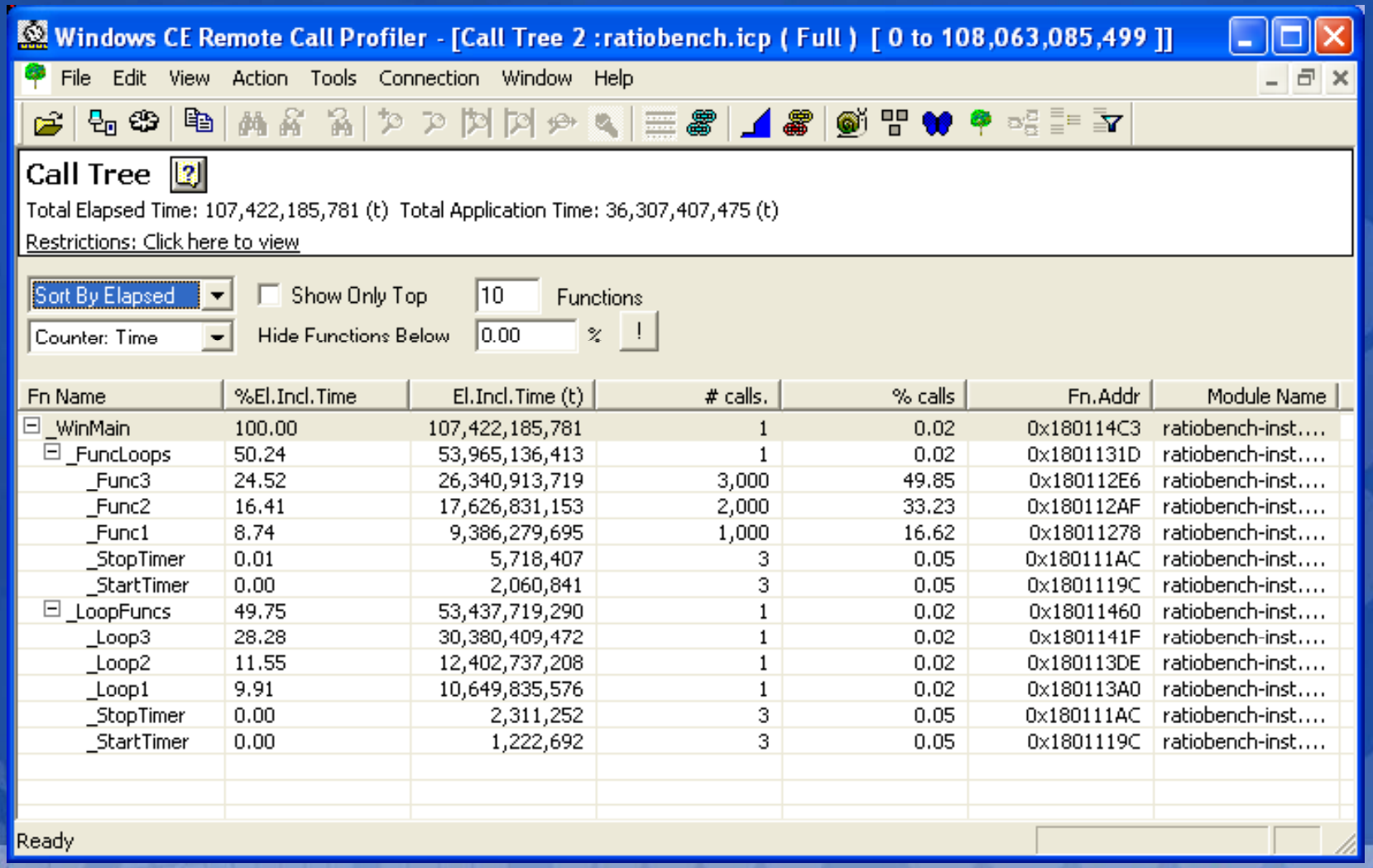
DEMO

Using Remote Kernel Tracker

Profiling function call

- PSL happens only in API call
- Normally function call won’t cause PSL
- Complier added function below to tracking function call
    - void __stdcall _CAP_Enter_Function (void *pfn);
    - void __stdcall _CAP_Exit_Function (void *pfn);

Profiling Code
Remote Call Profiler



Fn Name	%El.Incl.Time	El.Incl.Time (t)	# calls.	% calls	Fn.Addr	Module Name
_WinMain	100.00	107,422,185,781	1	0.02	0x180114C3	ratiobench-inst....
_FuncLoops	50.24	53,965,136,413	1	0.02	0x1801131D	ratiobench-inst....
_Func3	24.52	26,340,913,719	3,000	49.85	0x180112E6	ratiobench-inst....
_Func2	16.41	17,626,831,153	2,000	33.23	0x180112AF	ratiobench-inst....
_Func1	8.74	9,386,279,695	1,000	16.62	0x18011278	ratiobench-inst....
_StopTimer	0.01	5,718,407	3	0.05	0x180111AC	ratiobench-inst....
_StartTimer	0.00	2,060,841	3	0.05	0x1801119C	ratiobench-inst....
_LoopFuncs	49.75	53,437,719,290	1	0.02	0x18011460	ratiobench-inst....
_Loop3	28.28	30,380,409,472	1	0.02	0x1801141F	ratiobench-inst....
_Loop2	11.55	12,402,737,208	1	0.02	0x180113DE	ratiobench-inst....
_Loop1	9.91	10,649,835,576	1	0.02	0x180113A0	ratiobench-inst....
_StopTimer	0.00	2,311,252	3	0.05	0x180111AC	ratiobench-inst....
_StartTimer	0.00	1,222,692	3	0.05	0x1801119C	ratiobench-inst....

DEMO

Using Remote Call Profiler

Summary

Make your device better!

嵌入式开发资源

- **Windows Embedded**中文官方网站
  http://www.microsoft.com/china/windows/embedded
- **.NET Micro Framework**
  http://msdn2.microsoft.com/zh-cn/embedded/bb267253.aspx
- **Microsoft Robotics Studio**
  http://msdn2.microsoft.com/zh-cn/robotics/default.aspx
- 微软嵌入式开发者论坛
  http://forums.microsoft.com/china/default.aspx?ForumGroupID=493&SiteID=15
- 微软中国嵌入式开发者博客
  http://blogs.msdn.com/yunxu/
- **Mike Hall**的博客
  http://msdn2.microsoft.com/zh-cn/embedded/ Aa731228.aspx

微软启动新一轮“免费重考计划”

您的潜力，我们的动力
Microsoft
微软（中国）有限公司

- 作为对微软认证学习者的支持与鼓励，微软公司于2007年9月 15日启动新一轮免费重考计划。
- 如果您以前曾参与过该项计划，那您或许已经了解到该计划将为您顺利通过微软认证考试带来有效的保障。在计划推行期内，如果您在考试前注册获得免费重考考试券，那当您首次考试未通过时，您将获得一次免费重考的机会。

微软启动新一轮“免费重考计划”

您的潜力，我们的动力
Microsoft
微软（中国）有限公司

- 有关活动详情，请访问：

http://www.microsoft.com/china/msdn/events/featureevents/2007/Secondshot.aspx

- 赶快规划您的认证学习计划吧，现在注册即可获得免费重考考试券！

获取更多MSDN资源

您的潜力，我们的动力

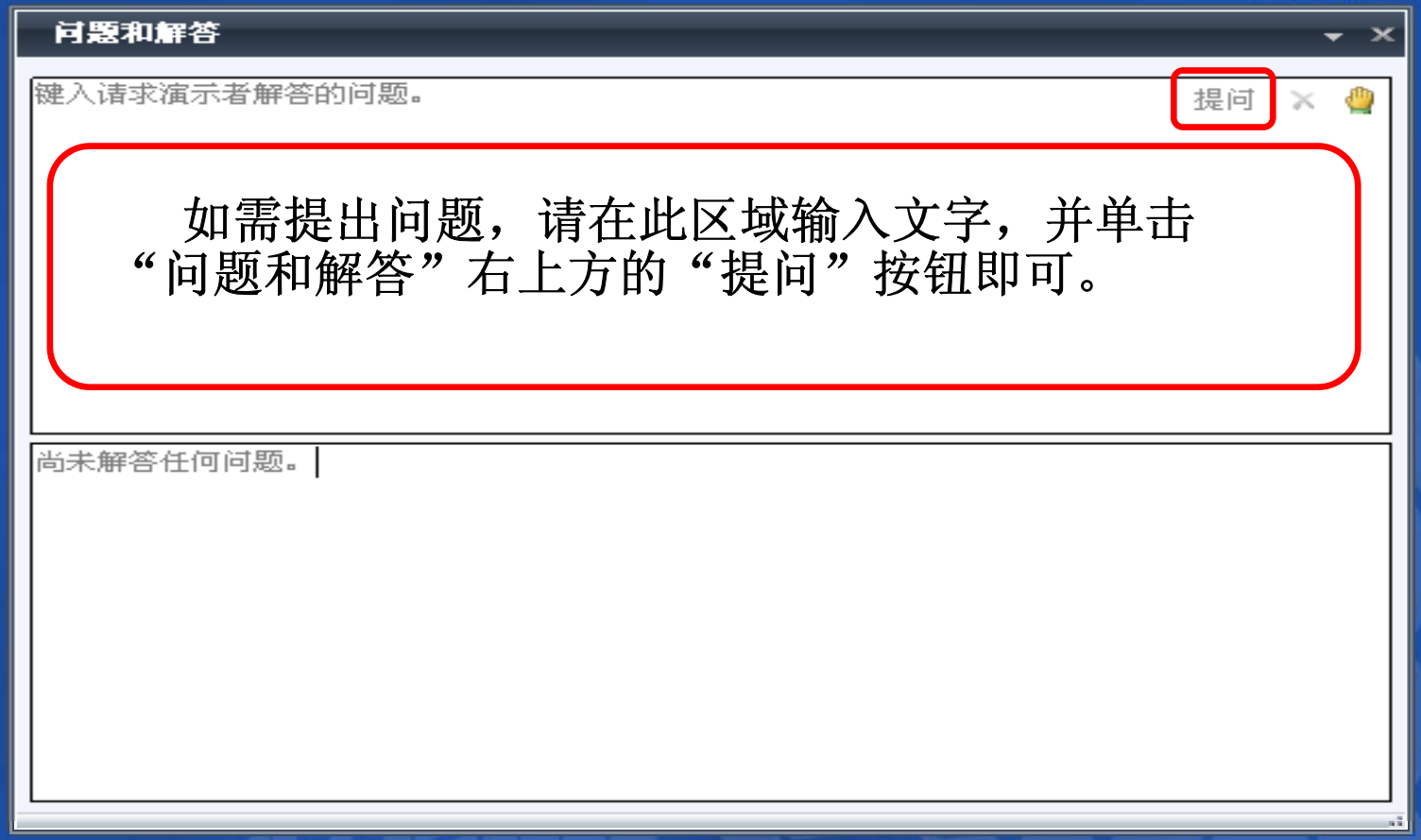
Microsoft
微软（中国）有限公司

- **MSDN中文网站**
  http://msdn2.microsoft.com/zh-cn
- **MSDN中文网络广播**
  http:// www.microsoft.com/china/msdn/webcast
- **MSDN免费中文速递邮件 (MSDN Flash)**
  http://msdn2.microsoft.com/zh-cn/flash
- **MSDN开发中心**
  http://msdn2.microsoft.com/zh-cn/developercenters
- MSDN图书中心
  http://www.microsoft.com/china/msdn/book

Question & Answer

您的潜力，我们的动力

Microsoft
微软(中国)有限公司



您也可以选择在微软中文技术论坛上寻求帮助，MSDN中文网络广播的讲师们会定期在论坛上为大家解答与课程相关的技术问题。
http://forums.microsoft.com/china


您的潜力，我们的动力
Microsoft
微软（中国）有限公司
Microsoft
msdn
MSDN Webcasts