

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 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



msdn

MSDN Webcasts

本次课程内容包括

您的潜力. 我们的动力

Microsoft
微软(中国)有限公司

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

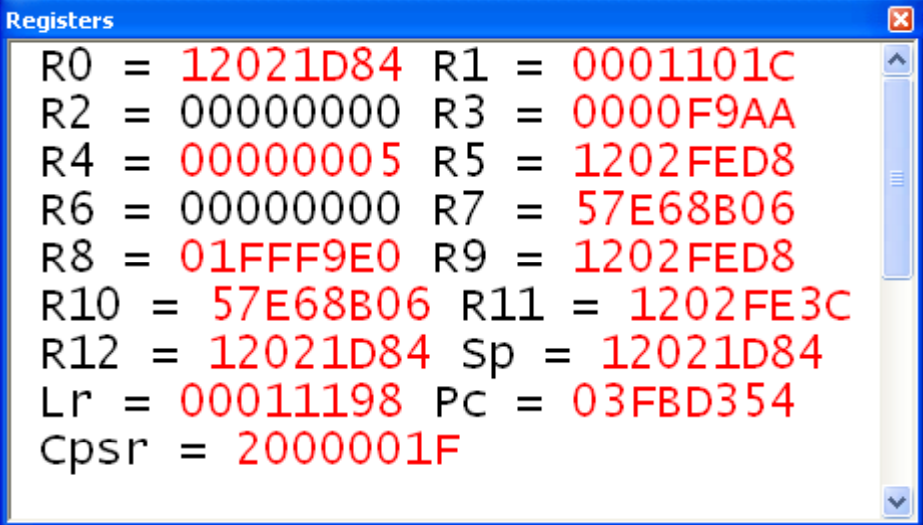
您的潜力. 我们的动力

Microsoft[®]
微软(中国)有限公司

CE and Mobile Overview

Memory Layout

Hint: Register Window

A screenshot of a 'Registers' window from a debugger. The window has a blue title bar with the text 'Registers' and a close button. The content area is white and displays the values of various registers. The registers are listed in two columns. The values are in hexadecimal. Some values are highlighted in red. The registers shown are R0 through R12, Lr, Pc, and Cpsr. The values are: R0 = 12021D84, R1 = 0001101C, R2 = 00000000, R3 = 0000F9AA, R4 = 00000005, R5 = 1202FED8, R6 = 00000000, R7 = 57E68B06, R8 = 01FFF9E0, R9 = 1202FED8, R10 = 57E68B06, R11 = 1202FE3C, R12 = 12021D84, Sp = 12021D84, Lr = 00011198, Pc = 03FBD354, Cpsr = 2000001F.

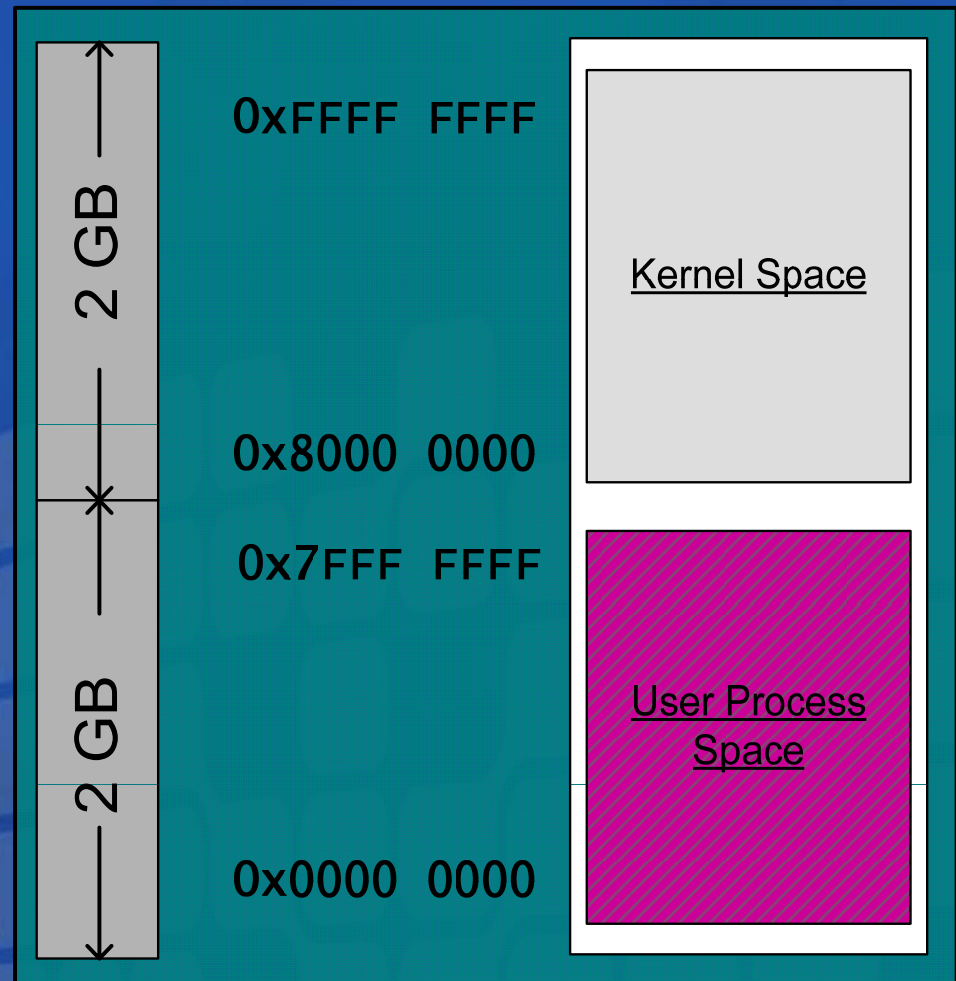
```
Registers
R0 = 12021D84 R1 = 0001101C
R2 = 00000000 R3 = 0000F9AA
R4 = 00000005 R5 = 1202FED8
R6 = 00000000 R7 = 57E68B06
R8 = 01FFF9E0 R9 = 1202FED8
R10 = 57E68B06 R11 = 1202FE3C
R12 = 12021D84 Sp = 12021D84
Lr = 00011198 Pc = 03FBD354
Cpsr = 2000001F
```

Windows CE Virtual Memory Review

您的潜力. 我们的动力

Microsoft
微软(中国)有限公司

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

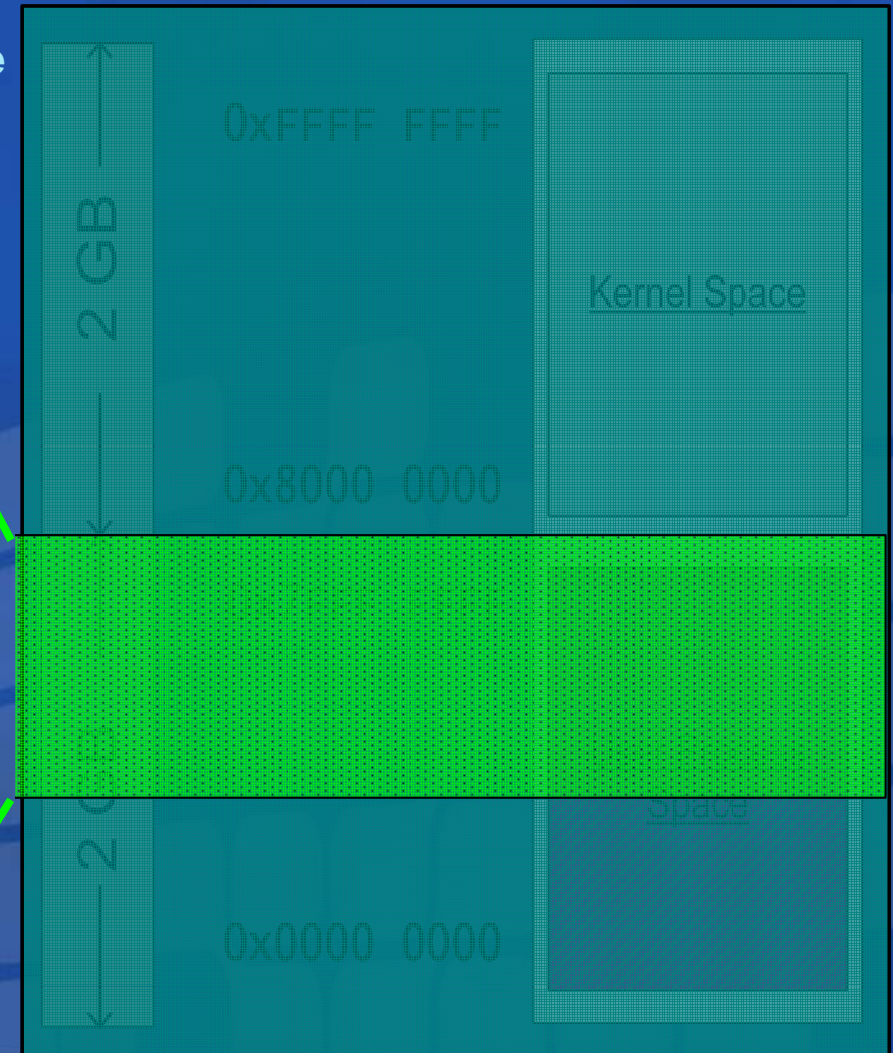
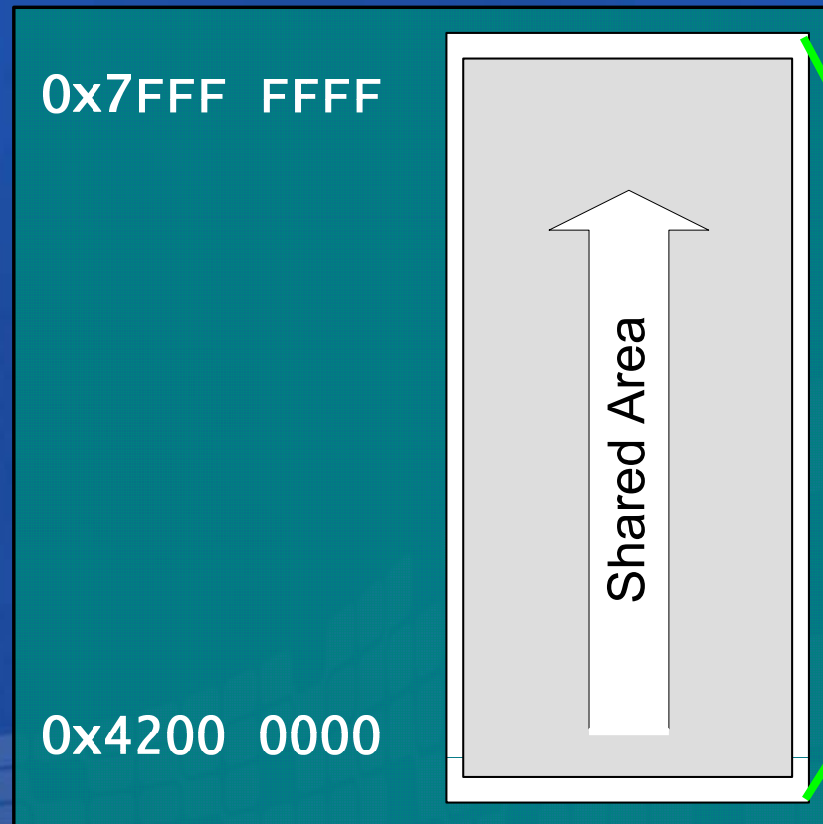


Windows CE Virtual Memory Review

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

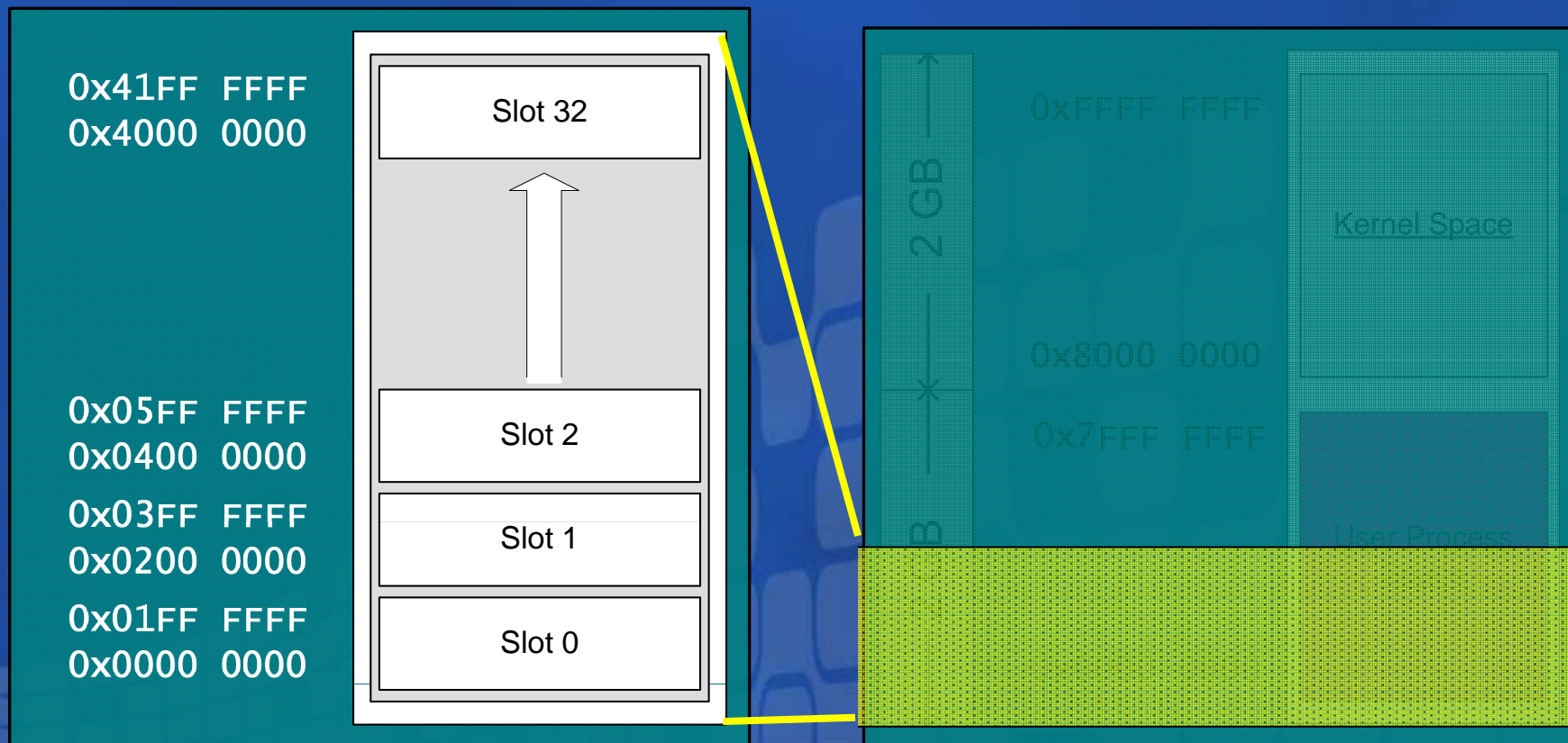


Windows CE Virtual Memory Review

32 MB Process slots take
the lowest addresses

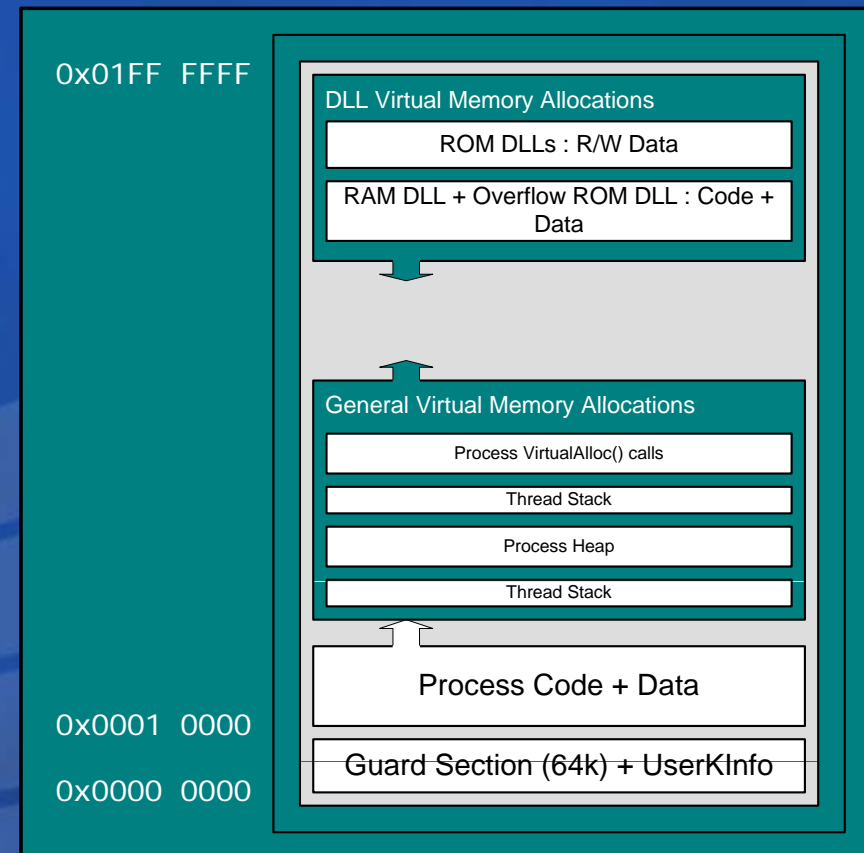
您的潜力. 我们的动力

Microsoft
微软(中国)有限公司



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



您的潜力. 我们的动力

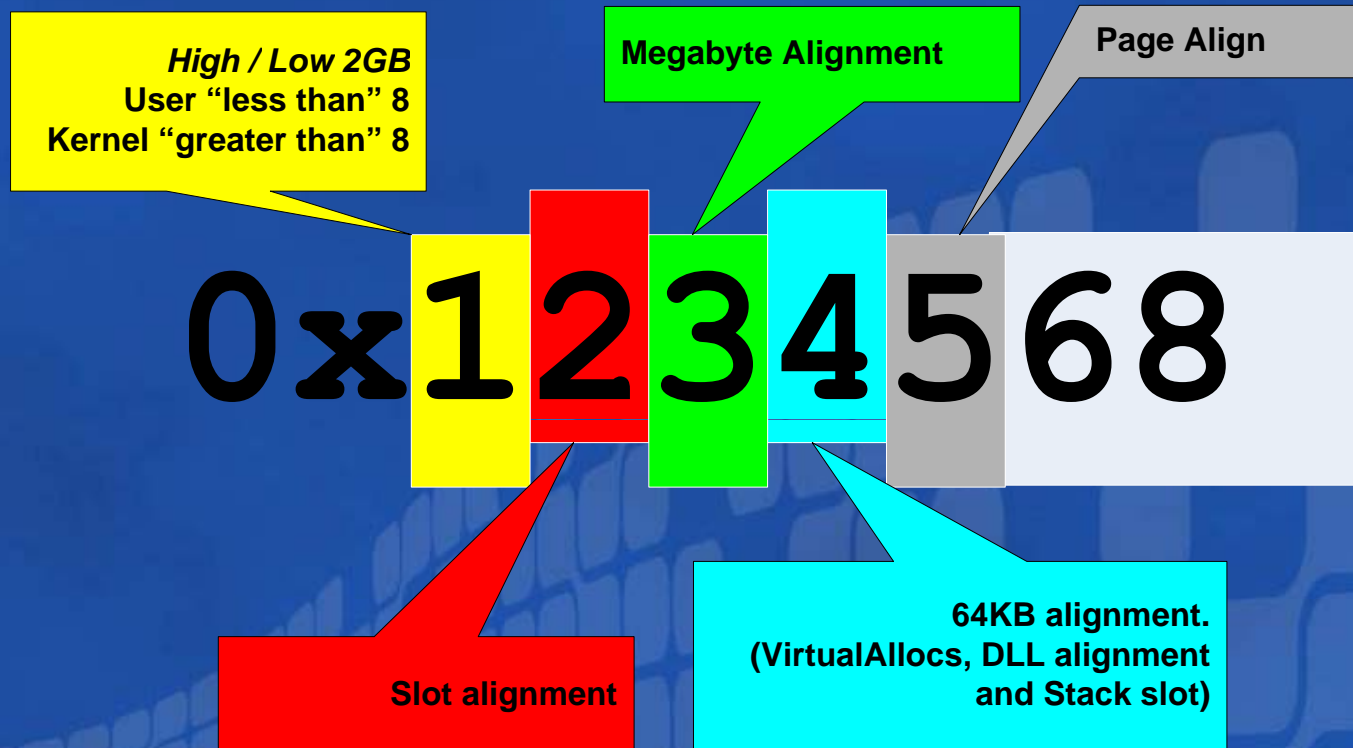
Microsoft
微软(中国)有限公司

Debugging tips

What do we learn from the memory layout?

Hex Address Anatomy

Hex addresses have identifiable positions:



Hex Address Anatomy

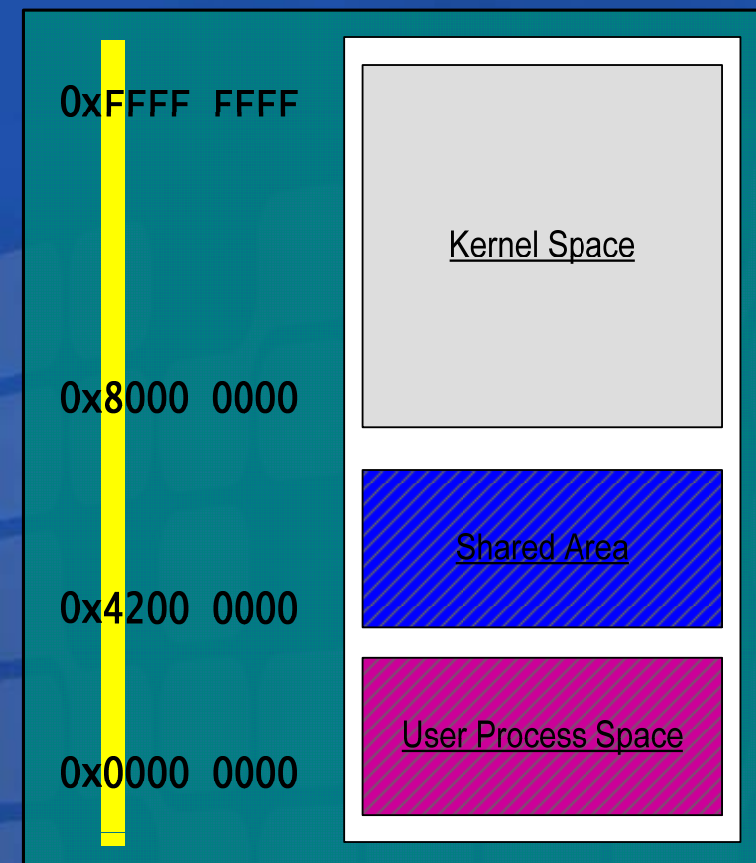
Position 1 (in yellow) tells you if its kernel
space or user space

8 and above is kernel

Below 8 is User space

High / Low 2GB
User "less than" 8
Kernel "greater than" 8

0x**1**2345678



Hex Address Anatomy

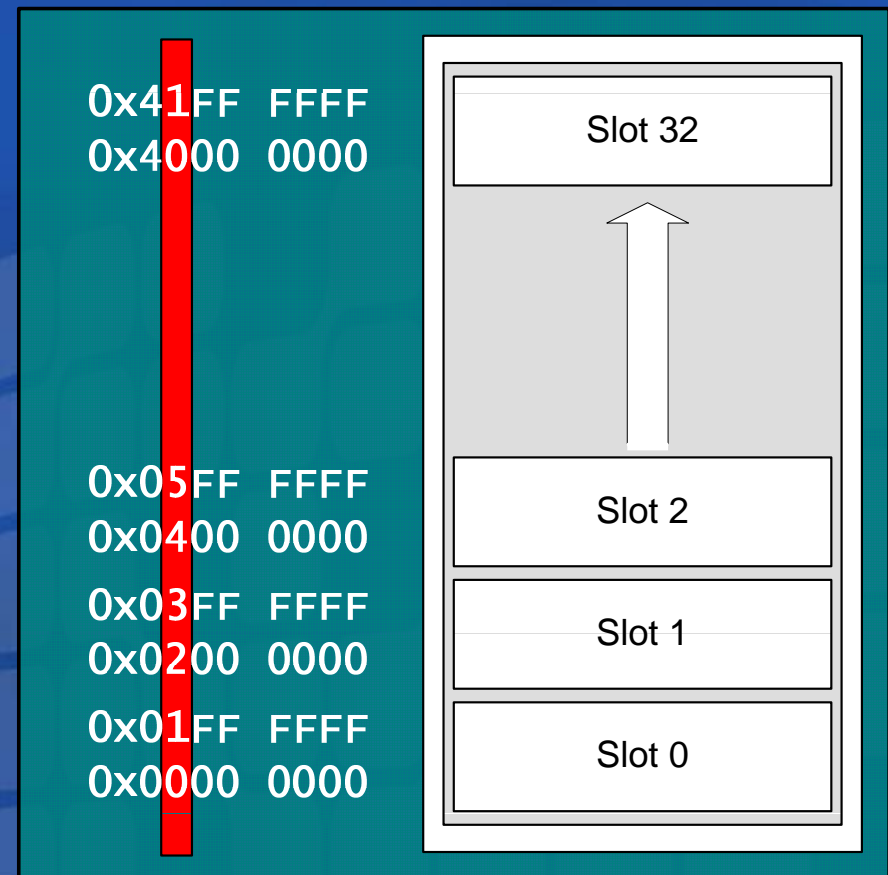
Position 2 (in red) is process slot position

If slot number is odd, address
is likely a DLL

If slot number is even,
address is likely an app

0x12345678

Process Slot alignment

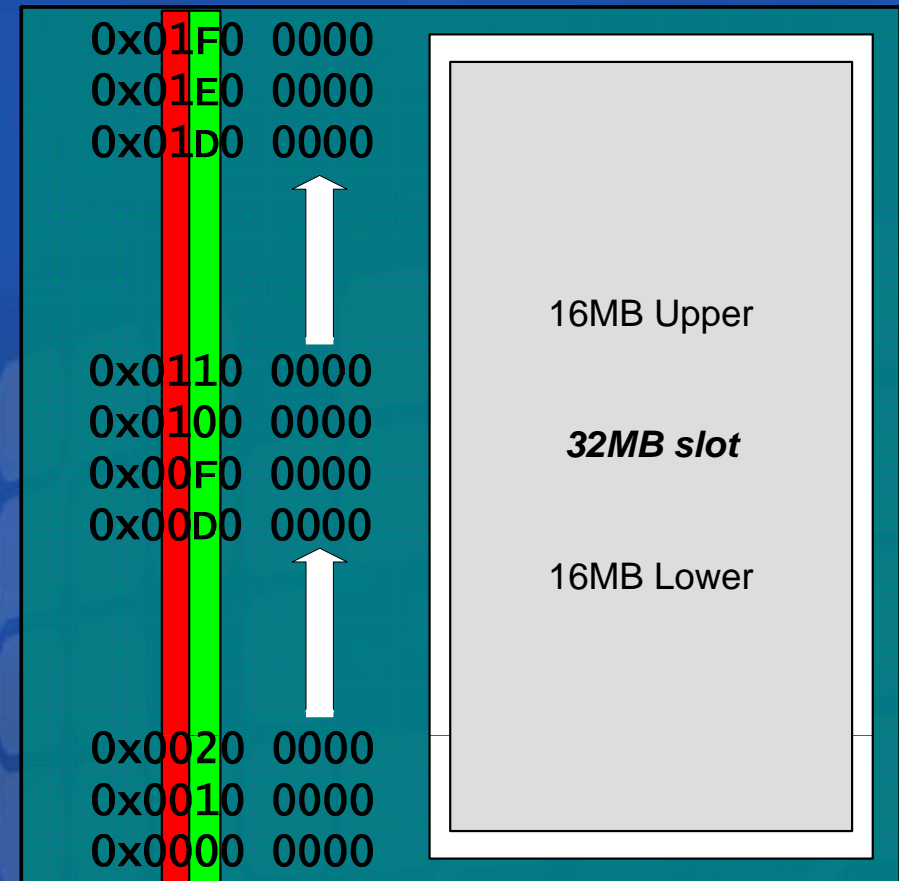


Hex Address Anatomy

32MB in hex is: 0x0200 0000

Megabyte Alignment

0x12345678



Hex Address Anatomy

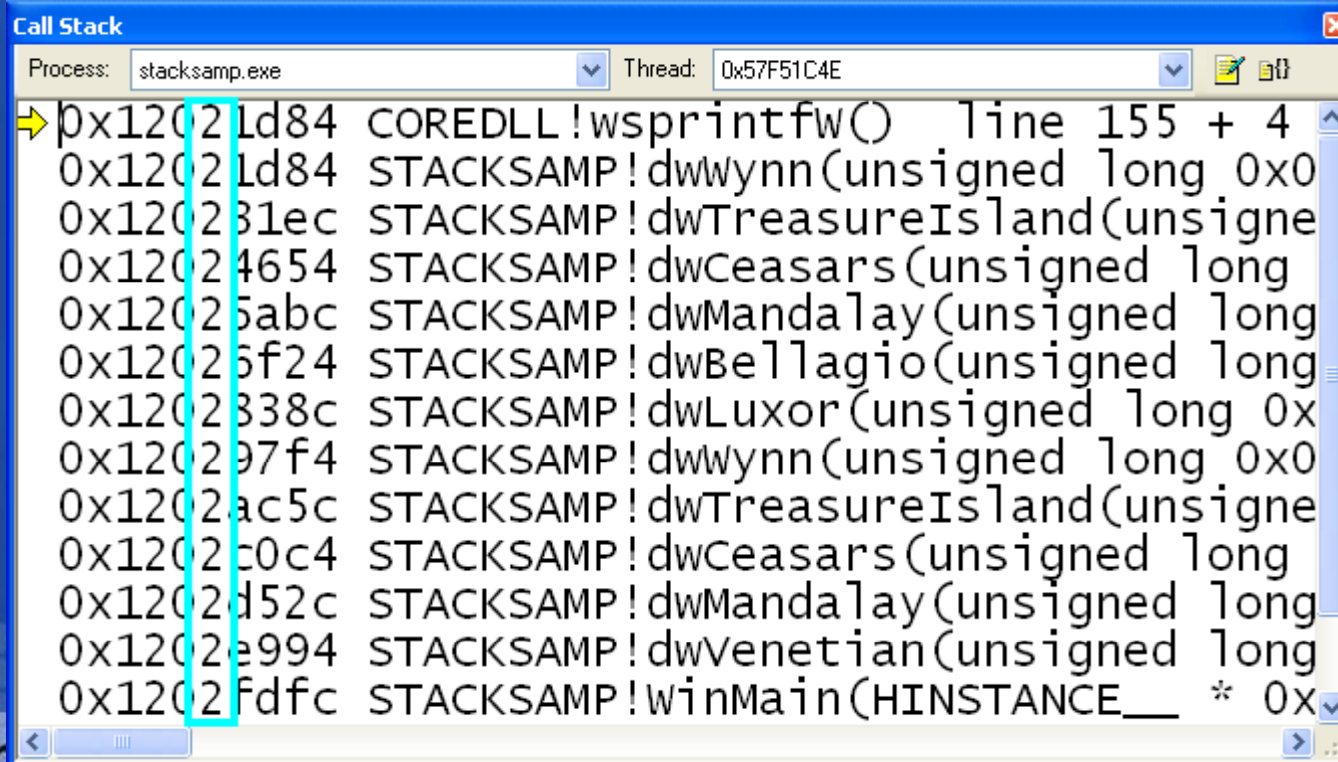
Position 4 (in cyan) is 64 kb alignment

DLL alignment

Stack alignment

0x12345678

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



```
Call Stack
Process: stacksamp.exe Thread: 0x57F51C4E
0x12021d84 COREDLL!wsprintfw() Line 155 + 4
0x12021d84 STACKSAMP!dwwynn(unsigned long 0x0
0x120231ec STACKSAMP!dwTreasureIsland(unsigne
0x12024654 STACKSAMP!dwCeasars(unsigned long
0x12025abc STACKSAMP!dwMandalay(unsigned long
0x12026f24 STACKSAMP!dwBellagio(unsigned long
0x1202838c STACKSAMP!dwLuxor(unsigned long 0x
0x120297f4 STACKSAMP!dwwynn(unsigned long 0x0
0x1202ac5c STACKSAMP!dwTreasureIsland(unsigne
0x1202c0c4 STACKSAMP!dwCeasars(unsigned long
0x1202d52c STACKSAMP!dwMandalay(unsigned long
0x1202e994 STACKSAMP!dwVenetian(unsigned long
0x1202fdfc STACKSAMP!WinMain(HINSTANCE__ * 0x
```

Hex Address Anatomy

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

16 pages == 64kb

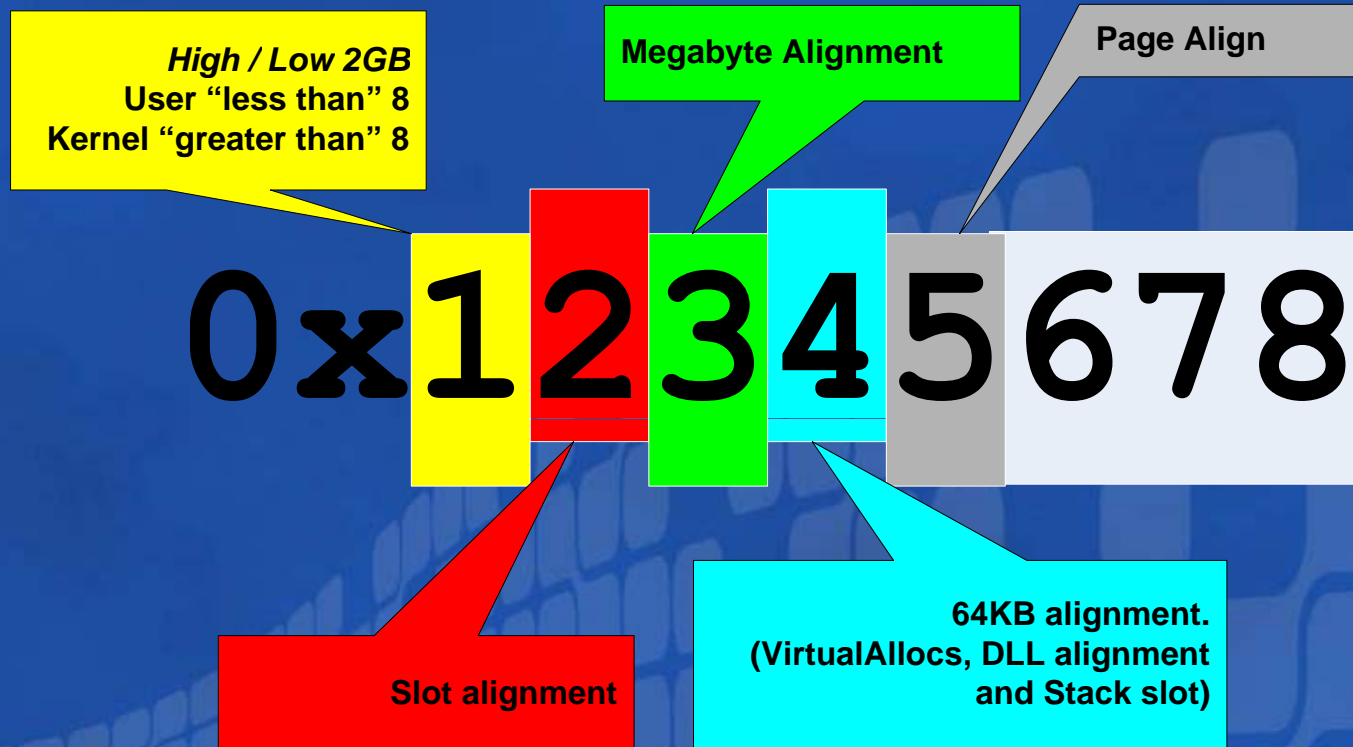
Page Align

0x12345678

```
Call Stack
Process: stacksamp.exe Thread: 0x57F51C4E
0x12021d84 COREDLL!wsprintfw() line 155 + 4
0x12021d84 STACKSAMP!dwWynn(unsigned long 0x0)
0x120231ec STACKSAMP!dwTreasureIsland(unsigned long)
0x12024654 STACKSAMP!dwCeasars(unsigned long)
0x12025abc STACKSAMP!dwMandalay(unsigned long)
0x12026f24 STACKSAMP!dwBellagio(unsigned long)
0x1202838c STACKSAMP!dwLuxor(unsigned long 0x)
0x120297f4 STACKSAMP!dwWynn(unsigned long 0x0)
0x1202ac5c STACKSAMP!dwTreasureIsland(unsigned long)
0x1202c0c4 STACKSAMP!dwCeasars(unsigned long)
0x1202d52c STACKSAMP!dwMandalay(unsigned long)
0x1202e994 STACKSAMP!dwVenetian(unsigned long)
0x1202fdfc STACKSAMP!WinMain(HINSTANCE__ * 0x)
```


Hex Address Anatomy

Lets take another look



Thinking in Hex Quiz 1

Microsoft
微软(中国)有限公司

- What is the difference between:

0x00119235

and

0x0A119235 ??

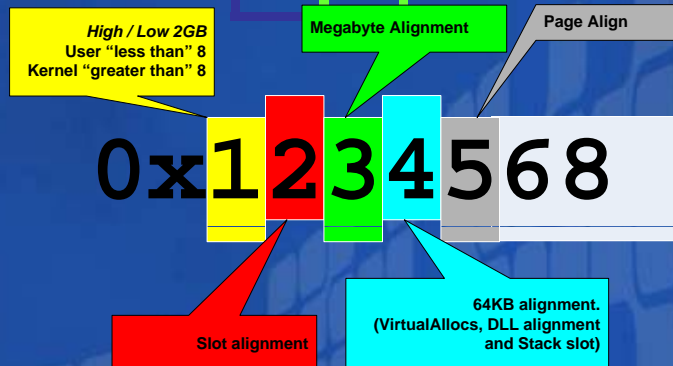
- Extra credit: What do you think it points to?

Thinking in Hex (Answer 1)

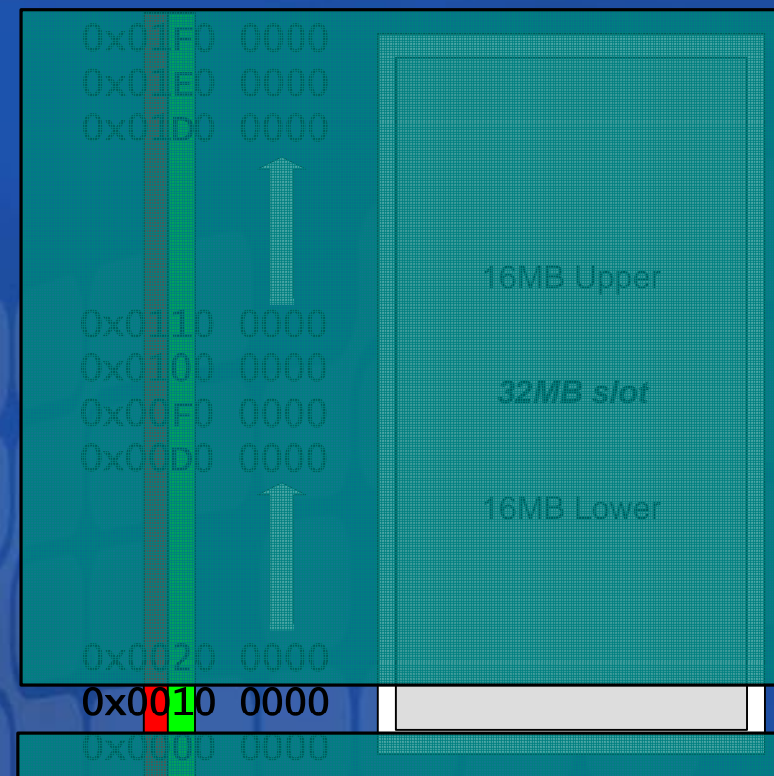
Same address!

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.

0x019ABCDE

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

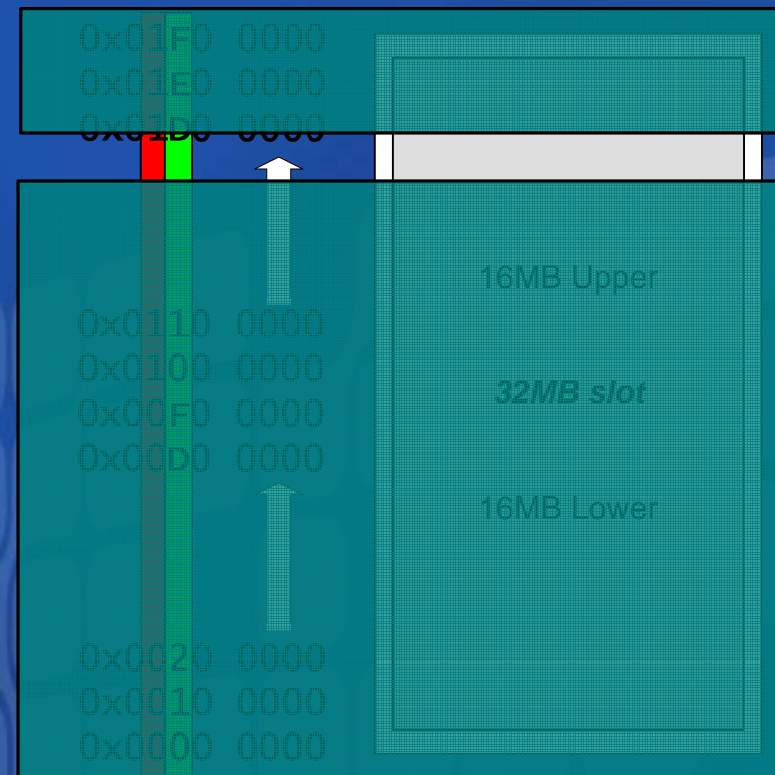
Megabyte Alignment

Page Align

0x1234568

Slot alignment

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



Visually Debugging Call Stacks

您的潜力. 我们的动力

Microsoft
微软(中国)有限公司

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

Stack Pointer “dipped” into first guard page

Stack overflow

Registers

R0 = 12021D84	R1 = 0001101C
R2 = 00000000	R3 = 0000F9AA
R4 = 00000005	R5 = 1202FED8
R6 = 00000000	R7 = 57E68B06
R8 = 01FFF9E0	R9 = 1202FED8
R10 = 57E68B06	R11 = 1202FE3C
R12 = 12021D84	Sp = 12021D84
Lr = 00011198	Pc = 03FBD354
Cpsr = 2000001F	

Call Stack

Process: stacksamp.exe Thread: 0x57F51C4E

0x12021d84	COREDLL!wsprintfw()	lin
0x12021d84	STACKSAMP!dwWynn(unsigned	
0x120231ec	STACKSAMP!dwTreasureIsLa	
0x12024654	STACKSAMP!dwCeasars(unsi	
0x12025abc	STACKSAMP!dwMandalay(unsi	
0x12026f24	STACKSAMP!dwBellagio(unsi	
0x1202838c	STACKSAMP!dwLuxor(unsign	
0x120297f4	STACKSAMP!dwWynn(unsigne	
0x1202ac5c	STACKSAMP!dwTreasureIsLa	
0x1202c0c4	STACKSAMP!dwCeasars(unsi	
0x1202d52c	STACKSAMP!dwMandalay(unsi	
0x1202e994	STACKSAMP!dwVenetian(unsi	
0x1202fdfc	STACKSAMP!winMain(HINSTA	

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

您的潜力. 我们的动力

Microsoft[®]
微软(中国)有限公司

DEMO

Target Control Debugging

您的潜力，我们的动力

Microsoft
微软(中国)有限公司

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 ?

您的潜力. 我们的动力

Microsoft
微软(中国)有限公司

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

Microsoft
微软(中国)有限公司

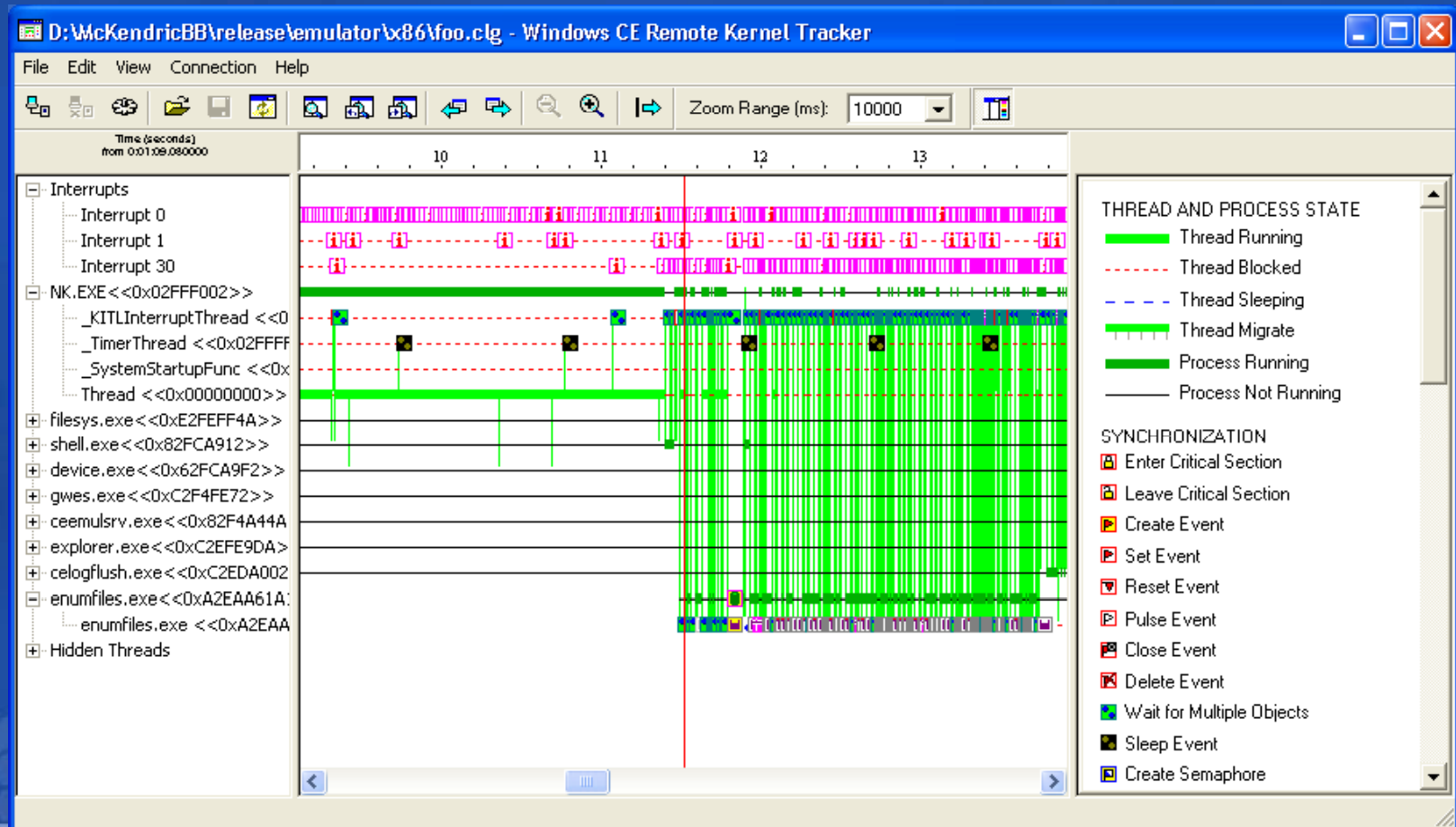
- Capture OS events as they happen
 - Thread and process creation/execution/PRIO
 - Synchronization objects
 - Interrupts
 - Memory, paging, TLB misses
 - Loader events
 - Boot events

Examining Thread Behavior

Remote Kernel tracker

您的潜力. 我们的动力

Microsoft®
微软(中国)有限公司



您的潜力. 我们的动力

Microsoft[®]
微软(中国)有限公司

DEMO

Using Remote Kernel Tracker

Profiling function call

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

您的潜力，我们的动力

Microsoft®
微软(中国)有限公司

Windows CE Remote Call Profiler - [Call Tree 2 :ratiobench.icp (Full) [0 to 108,063,085,499]]

File Edit View Action Tools Connection Window Help

Call Tree

Total Elapsed Time: 107,422,185,781 (t) Total Application Time: 36,307,407,475 (t)

Restrictions: [Click here to view](#)

Sort By Elapsed ☐ Show Only Top 10 Functions

Counter: Time Hide Functions Below 0.00 % !

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....

Ready

您的潜力, 我们的动力

Microsoft[®]
微软(中国)有限公司

DEMO

Using Remote Call Profiler

您的潜力. 我们的动力

Microsoft
微软(中国)有限公司

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>

您的潜力. 我们的动力

微软启动新一轮“免费重考计划”

Microsoft®
微软(中国)有限公司

认证自己, 成就未来

微软推出“免费重考”计划



- 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。

您的潜力, 我们的动力

微软启动新一轮“免费重考计划”

Microsoft
微软(中国)有限公司

认证自己, 成就未来

微软推出“免费重考”计划



• 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

- **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](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

问题和解答

键入请求演示者解答的问题。

提问

如需提出问题，请在此区域输入文字，并单击“问题和解答”右上方的“提问”按钮即可。

尚未解答任何问题。 |

您也可以选择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