Windows® 2000 64-Bit

Jeff Havens
Architect
Windows NT Base Systems
Microsoft® Corporation
Agenda

- Overview
- Development Environment
- System design
- Call-to-Action
- Collateral
Windows 2000 64-Bit Goals

- Porting from Win32® to Win64™ should be simple
- A single source code base for Win32® and Win64™
- Enable existing applications to scale to enterprise capacities
- Enable new designs that use huge address space and memory
What Is 64-Bit Version Of Windows 2000?

- 64-Bit Windows is an evolution of the Windows programming model and APIs
- 64-Bit Windows 2000 is Window 2000 expanded to 64 bits
- Pointers are expanded to 64 bits
- Memory allocation sizes are 64 bits
Product Description

- Windows 2000 64-Bit is basically Windows 2000
  - IIS
  - Active directory
  - PNP/Power management
- With some legacy support removed such as: ISA cards, 16 bit windows
Project Accomplishments
And Next Step

- 1/97 - Project design and prototype
- 1/98 - Design preview
  - First Win64™ “SDK”
- 3/98 - Win64™ kernel booted
- 4/98 - WinHEC
  - First Win64™ “DDK” and second Win64™ “SDK”
- 4/99 - Intel SDK 0.6
- 4/99 – WinHEC’99 demo
- 3Q’99 Itanium™ hardware boot
- 9/99 Intel SDK 1.7
- Beta 1H’00
Microsoft Applications plans for Win64™

- SQLServer
- Exchange
- Backoffice
- Office will run as IA-32 Windows 2000 application
Tools for Itanium™

- Visual Studio: 64-bit Edition
  - C++ FE w support for Win64™ migration
  - Optimizing Backend, tuned for Itanium™
  - Linker
  - CRT
  - MFC/ATL
  - Native Debugger
  - Visual Basic
  - Integrated Development Environment
  - Wizards, etc.
World-class performance delivered through advanced optimizations such as:

- Predication/speculation support
- Software Pipeliner
- Global scheduler
- Memory hierarchy optimizations
- Branch optimizations
- Advanced loop unrolling
- Profile Guided Optimization
- Whole Program Mode optimizations
- Object oriented optimizations
VC++ Compiler Status

- **Current status of the compiler**
  - Currently used to compile Windows 2000 with full optimization
  - Self-hosted on simulator
  - Compiles entire MS Office source base
Win64™ Abstract Model

- Win32® APIs and program model
- Adds new explicitly sized types
- Adds new integral types that match the precision of a pointer
- Pins the sizes of the major Windows 2000 and Windows types for both Win32® and for Win64™
- Almost all Win32® 32-bit data types remain 32-bits
- Pointers, LPARAM, WPARAM, LRESULT, HMODULE are 64-bits
Win64™ API Set

- Simple pointer stretch port of Win32® (and Windows 2000 Native) API set
- Win64™ data type definitions define most of the port
- Porting Issues are
  - Polymorphic Data usage
  - Pointer/length combinations
  - Miscellaneous cleanup
  - Cross 32/64 bit process communication
Image Format

- PE32+ (magic number 0x20b)
- SizeOfImage same as PE32 (limited to 4GB)
- ImageBase widen to 64 bits
- Stack sizes widen to 64 bits
- Heap sizes widen to 64 bits
- Import Address table entries are 64 bits
- Text sections are position independent
Memory Management

- Major redesign
- Three level page tables
- Self mapping
- Virtual page numbers are 64 bits - $\log_2(\text{PAGE\_SIZE})$
- Page table entries are 64 bits
- Physical memory addresses are 64 bits
Memory Management IA-64

- Currently 8 kilobyte page size
- Two sets of page tables, system/user
- One system root page table
- One user root page table per process
- $2^{44} = 2 \times 2^{10} \times 2^{10} \times 2^{10} \times 2^{13} = 16 \text{ TB}$
- LOWEST\_SYSTEM\_ADDRESS > 0
Memory Management
System Areas

- Eight terabyte kernel
- One terabyte system cache
- Eight gigabyte HyperSpace
- 128 gigabyte paged pool
- 128 gigabyte System PTE space for kernel threads, etc.
- 128 gigabyte non-paged pool
Drivers

- Native drivers only
- Drivers need to be PNP
- The DDK model for drivers is unchanged from Windows 2000
- Some drivers will need to support 32 and 64 version of Ioctls
- I/O request length are limited to 32 bits
- Supply Microsoft the source for drivers
General Code Changes

- Cast pointers to PCHAR for +/-
  - `ptr = ((PVOID) (PCHAR) ptr + pageSize);`
- Cast pointer to LONG_PTR for masking
  - `ptr = (PVOID) ((LONG_PTR) ptr & -8);`
  - `~((UINT64) (PAGE_SIZE - 1)) != (UINT64)`
- `sizeof(P1 - P2) == sizeof(ptrdiff_t) == 8`
- Use `%l` to print `xxx_PTR`
- Compiler/CRT `size_t` is 64-bits
  - Sizeof, new, malloc, strlen etc.
- CRT `time_t` is 64 bits
Areas to consider

- Explicit and implicit unions with pointers
- Data structures stored on disk or exchanged with 32 bit processes
  - Security descriptors
- Code which uses the high address bit
- Functions with pointers as out parameters
  - BOOL GetBuf ( int fd, ULONG_PTR *buf);
- Code which deals with region sizes
Areas to consider

- **Piecemeal size allocations:**

```c
struct foo {
    DWORD NumberOfPointers;
    PVOID Pointers[1];
} xx;

Wrong:
malloc(sizeof(DWORD)+100*sizeof(PVOID));

Correct:
malloc(offsetof(struct foo, Pointers) +100*sizeof(PVOID));
```
ISV/IHV Enabling Programs

- Win64™ SDK/DDK as part of Windows 2000 SDK/DDK
- MSDN™
- Intel events
- Microsoft events
- Industry events
Additional Information

- Read “Getting Ready for 64-bit Windows” porting guide on Windows 2000 SDK
- Send feedback and questions to nt64feed@microsoft.com
Call To Action

- Make sure your application is Windows 2000 ready
- Prepare for Win64™ now
- Find problem areas
  - Start to design them out of system
- Install the Windows 2000 SDK and readme64.txt
  - Use types and functions defined in <basetsd.h>
Call To Action

- Remove pointer truncations
- Correct your polymorphism
- Compile warning free
- Ensure plug-in interfaces are RPCable
- Make COM objects runnable out of process
- Consider ways to use the larger address space and higher performance