Porting for Windows® on IA-64

Stan (Stawsh) Murawski
& Kyle Marsh
Developer Relations Group,
Shahrokh Mortazavi
Visual Languages & Tools,
Microsoft Corporation

Windows is a registered trademark of Microsoft Corporation in the United States and/or other countries.
Agenda

- 64-bit Windows® Overview.
- 64-bit Windows® Applications.
- Win64 types and API.
  - Making Win32 code 64-bit ready.
- Development Demo.
- Tools Roadman for 64-bit Windows®
  - MS Windows Platform SDK & Visual Studio.
- Other Issues and Call to Action.
64-bit Windows®

- 64-bit Windows is Windows®, i.e.,
  - Feature set equivalent to Windows 2000.
- Win64™ API, “same as” Win32™ except ...
  - All pointers, passed and returned, are now 64-bit pointers.
- WOW64 runs Win32 Applications.
  - Win32 (IA-32) x86 application binaries run.
  - Win16 applications do not run.

Features Equivalent to Windows 2000
64-Bit Windows Advantage

1. Larger virtual memory.
   - 4TB user and 4TB kernel.

2. Supports 64 bit integers.
   - Count more stuff in _int64 types.
   - Count faster with native 64 bit math, v.s. crt routines or user developed algorithms that use multiple “small” 32 bit registers.
   - Itanium™ Processor FPU counts faster too.

3. IA-64 code can reach performance potential of Itanium™ processors
64-bit Candidate Apps

1. Applications that use very large objects,
   - i.e., larger than 2 gigabyte objects.
   - e.g., Video editing.

2. Apps that use files that are larger than $2^{32}$ bytes.
   - You can memory map very LARGE files!
   - e.g., databases.
64-Bit Candidate Apps (cont)

3. Apps that count > $2^{32}$ “things”.
   – Statistical analysis applications.
   – _Int64 algorithms, e.g. 40 or 56 bit encryption “in a register”.
     (128 bit in 2 registers).

4. Apps HW Performance constrained.
   – Access to memory.
   – I/O and Bus speed.
MS 64-bit Windows®
Candidate applications

- MS SQL Server™
  - Very large things
- MS Exchange™ Server
  - Very many things
- MS BackOffice™ Servers
  - Large virtual and physical memory

- But not all apps:
  - Microsoft Office™ will run as a Win32 (IA-32) application.
64 bit Windows types, type Rules and API
WORDS, INTs, Pointers 64

• Size Model is called LLP64.
  – Default INTs and LONGs are 32 bits.
  – Pointers are 64 bits.
  – For 64 bits integers use, e.g., _int64.

• LLP64 goodness for developers.
  – Maximum Windows ISV application compatibility. IA-64 benefits sooner.
  – Most compatible Win32-Win64 interop.
    • Least change to data structures.

LLP64 is least change to Win32 code
Win64 types

• New explicitly sized types.
  • DWORD32, INT64.

• New integral types that match the precision of a pointer.
  • DWORD_PTR.

• Some 32-bit Win32 data types 64 bits:
  – Pointers are 64 bits, plus LPARAM, WPARAM, LRESULT, HMODULE.

Most Win32 32-bit types remain 32 bit
<table>
<thead>
<tr>
<th>TYPE NAME</th>
<th>WHAT IT IS</th>
</tr>
</thead>
<tbody>
<tr>
<td>LONG32, INT32</td>
<td>32-Bit Signed</td>
</tr>
<tr>
<td>LONG64, INT64</td>
<td>64-Bit Signed</td>
</tr>
<tr>
<td>ULONG32, UINT32,</td>
<td>32-Bit Signed</td>
</tr>
<tr>
<td>DWORD32</td>
<td></td>
</tr>
<tr>
<td>ULONG64, UINT64,</td>
<td>64-Bit Signed</td>
</tr>
<tr>
<td>DWORD64</td>
<td></td>
</tr>
<tr>
<td>TYPE NAME</td>
<td>WHAT IT IS</td>
</tr>
<tr>
<td>-------------------</td>
<td>--------------------------</td>
</tr>
<tr>
<td>INT_PTR, LONG_PTR</td>
<td>Signed Int, Pointer precision</td>
</tr>
<tr>
<td>UINT_PTR, ULONG_PTR, DWORD_PTR</td>
<td>Unsigned Int, Pointer precision</td>
</tr>
<tr>
<td>SIZE_T</td>
<td>Unsigned count, Pointer precision</td>
</tr>
<tr>
<td>SSIZE_T</td>
<td>Signed count, Pointer precision</td>
</tr>
</tbody>
</table>
Win64 type “Rules”

- For integral pointer types:
  - use UINT_PTR, INT_PTR, ULONG_PTR, or DWORD_PTR.
  - Do not assume that DWORD, LONG or ULONG can hold a pointer.

- Use SIZE_T to specify byte counts that span the range of a pointer.

- Make no assumptions about the length of a pointer or xxxx_PTR or xSIZE_T.
  - Assume these are compatible precision.
The Win64™ API

- Simple pointer stretch of Win32® (and NT Native) API set.
- Win64™ data type definitions in basetsd.h define most of the change.
- Primary Issues are:
  - Polymorphic Data usage, e.g., use of (DWORD / PSTR).
  - Pointer/length combinations.
  - Miscellaneous cleanup, e.g, (0xFFFFFFFF for -1).
- Cross 32/64 bit process communication.
Making code “64 bit ready”

Compile warning free!
Code Areas to Review (1 of 3)

- Code which uses the high address bit.
- Pointer truncations.
- Functions with pointers as out params.
  - `BOOL GetBuf( int fd, ULONG_PTR *buf);`
- Explicit and implicit unions with pointers.
- Data structures stored on disk or exchanged with 32 bit processes.
  - Structures that contain the types that change size, e.g., `LPARAM`, `WPARAM`, `LRESULT`, `HMODULE`.
Code Areas to Review (2 of 3)

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));
```
Code Areas to Review (3 of 3)

- Correct reference to polymorphic data.
- Ensure plug-in interfaces are RPC-able.
- Make COM objects able to run out of process.
- All assembly code.
  - It’s not x86 (IA32) assembler.
Demos:
Win64 code and
Win64 build
Demo:

Debugging Win64 code
Win64 Device Drivers

- Device Drivers are 64 bit code.
  - No support for 32 bit device drivers.
  - At IDF: “Win64 Device Driver Porting”.
- Code signing “safety rules” same as Win32.
- Drivers need to be PNP.
- Drivers need to consider IF they will be called from 32 and 64 bit mode code.
  - need to support 32 and 64 version of IOCTLs.
  - I/O request length is limited to 32 bits.

Drivers must be 64-bit, PNP & signed
Win64 Rapid migration

● Situation
  – You want to be IA-64.
  – 2 gig address space is AOK enough.
  – LOTs of Pointer truncation warnings.
  – Pointers and int/long are freely mixed.
  – Polymorphism via 32-bit types is used heavily.

● Alternative – Run in a 32-bit 64-bit address space “sandbox”.

22
Address Space “sandbox”

- **IMAGE_FILE_LARGE_ADDRESS_AWARE**
  - If SET, 64-bit address space available.
  - If CLEAR set, never/can’t see > 2GB.
    - Upper 33 address bits are 0.
    - Can truncate 64 bits, and extend 32 bits.

- **Example OK Code:**
  ```c
  DWORD dw;
  PVOID dest, src = malloc(IO_BUFFER);

  dw = (DWORD)src;
  dest = (PVOID)dw;
  ASSERT(((DWORD_PTR)src & 0xffffffff80000000) == 0);
  ASSERT(src == dest);
  ```
Win32 on 64 bit Windows

- Address space is either 64 or 32.
  - Can **not** mix 32 and 64 in an address space

- 64-bit processes run Win64 APIs which call directly into the 64-bit kernel.

- 32-bit processes run Win32 APIs using 32-bit ntdll, kernel32, user32, etc.
  - 32/64 thunk made at System-Call interface between user-mode and kernel-mode.
    - Provides excellent compatibility due to small, validated, strictly defined API set.

No mixing 64/32 bit code in same process
64-bit Windows roadmap

<table>
<thead>
<tr>
<th>Year</th>
<th>Q4</th>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
<th>Q1</th>
<th>Q2</th>
</tr>
</thead>
<tbody>
<tr>
<td>2000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2001</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **SDK 2.0**: OS ships with Intel Cross Dev. SDK
- **BETA**: MS BETA of 64 bit Windows
- **RTM**: 64 bit Windows RTM

Itanium™ & 64 bit Windows RTM
64-bit Windows roadmap

- Early HW developer bits.
- Joint Intel/MS SDK:
  - Cross-Dev SDK (Build#5), Jan’00
    - RTM Windows 2000 code base.
  - SDK2.0, Mar’00
- MS 64-bit Windows BETA1
  - Spring/2Q ’00.
  - Platform SDK (tools&OS) ships & supported by MS
- 64-bit Windows RTM (ship)
  - When computers with the Itanium™ Processor ship.
Tools Roadmap
For
64-bit Windows

Shahrokh Mortazavi
Lead Program Manager
Visual Languages & Tools
Microsoft

February 15-17, 2000
Agenda

- Visual Studio Roadmap
- SDK Tools Roadmap
- Tools status
- Optimization features
- Futures
- Call to Action
SDK Tools roadmap

- **2000**
  - Q4: Itanium™ & 64-bit windows RTM
  - Q1: PSDK0
  - Q2: PSDK2
  - Q3: MS Platform SDK + tools
  - Q4: Release candidate

- **2001**
  - Q1: Simulator based, core tools only
  - Q2: Targets real HW; debugger added
SDK tools roadmap

• Joint Intel/MS
  – Cross Dev SDK (build#5), Jan’00
    – Compiler FrontEnd from next major release of VC
    – Backend/Optimizer
    – Linker, MFC, ATL
  – SDK 2.0, Mar’00
    – VC Debugger added; Improved code quality; bug fixes

• Future PSDKs (DDK) directly from MS
  – Non-NDA, free

Today: SDKs, Soon: PSDKs
SDK Tools status

- Robustness/Correctness
  - Have been compiling NT since last year
  - Compiler bootstrap, SQL server, VC language tests
    - All with optimizations turned on
  - SDK user: “compiled/linked 10M LOC without problem”

- Code Quality
  - Parallel Itanium Dev team
  - All major optimization phases implemented
  - Currently focused on tuning

SDK Tools are here & usable today!
Visual Studio Tools Roadmap

• **Visual Studio next major release**
  - 32-bit, focused on Enterprise/Web development
  - Various Intel specific features for KNI, WNI, 64-bit migration.

• **64-bit Visual Studio**
  - Same as above, ported and tuned for Itanium™
    - Cross tools vs. Native tools
  - Mature optimization technology
  - Advanced debugging support: debug of optimized code!
Optimization specifics

- Partial list of optimizations:
  - Predication
  - Speculation
  - Local/global scheduling
  - Whole Program and Profile Guided Optimization
  - Software pipelining
  - All std opts: CSE, loop unrolling, branch opts, etc.

- Tuning focus
  - Respectable SPEC numbers: Not a “SPEC Warrior”
  - TPC benchmark
  - Integer, RWC performance is our primary focus

MS will deliver a world-class compiler
Profile-Guided Optimization

Compile:
- Source
- Compile: Insert instrumentation code
- Instrumented object

Run:
- Input
- Instrumented object
- Profile data
- Output

Re-Compile:
- Source
- Profile data
- Re-Compile: Use profile data
- Optimized object

A Critical Optimization for best performance!
Tools Futures

- Improved debug of optimized code
- Improved optimizations
- Better FP code generation
- More analysis tools
- Improved compiler throughput
- Continued collaboration with MS-Research
64-bit Windows Platform SDK

- Intel IA-64 SDK <= 1.7 included simulator.
  - Started work before IA-64 Hardware avail.
- Intel “Cross-Dev” SDK avail Jan ‘00.
  - For developers with IA-64 early Hardware.
- DDK available for Driver Developers.
MS Developer Programs

- MSDN™ Developer Programs.
  - See “Partnering” at http://MSDN.Microsoft.com/.

- MS Developer Relations Initiatives: mailto:DrgWin64@Microsoft.com
  - Report 3rd party dependencies, both COM server Components and DLLs.
  - non-support 64 bit questions/comments.

- MS 64-bit Windows Tech-BETA.
  - Apply at BetaInfo@Microsoft.com.

MSDN is a trademark of Microsoft Corporation in the United States and/or other countries.
Collateral and Feedback

  – Platform SDK CD (since Sept ’98).
  – http://msdn.microsoft.com
     MSDN Online Library
     see “Getting Ready for 64 bit Windows”

● Beta site: betainfo@microsoft.com

● mailto:nt64feed@microsoft.com
  64 bit Windows feedback / questions.

Tell us about “other” code that your 64-bit code depends upon to build/run!
Call To Action - Readiness

- Start with good code for Windows 2000.
  - Follow the Design Guidelines.
- Install the Windows 2000 Platform SDK.
  - “Know“ what is in readme64.txt.
  - Use `<basetsd.h>` types and functions.
- Get 64 bit ready now.
  - Design “problem areas” out of your code.
    - No pointer truncation.
    - Correct polymorphism.
  - Clean build Win32™ code for Win64™.

No Source Fork
64-bit ready NOW!
Call To Action – Exploit 64

- New Function – new design options.
  - You can do stuff with Big INTs.
    - Count a lot of stuff, w/o FP.
  - You can do things with Large memory.
    - Memory mapping HUGE files.
    - HUGE arrays and structures.

- More Speed and Scale (even existing Apps)
  - Optimize for 64 bit “size” (see above)
  - Optimize for IA-64

Consider how you can use large address spaces!