
# Delphi and 64 bit

Jeroen Pluimers
jpluimers@better-office.com
Edwin van der Kraan
evdkraan@better-office.com
better office benelux

EKON15 2011
DE, Düsseldorf, 20111028

## Agenda

- Introduction
  - x86-64 overview
  - Modern platform limits
- 32-bit vs. 64-bit
  - 32-bit code on 64-bit platform
  - Points to consider
  - Who needs 64-bit
- 64-bit data models
  - Data models overview
  - Delphi data model
  - Why SizeOf(Integer) = 4 on x86-64?
- Moving your application to x86-64:
  - Rules
  - Common issues
- DEMO

## Introduction - x86-64 overview

- x86-64 is an extension for IA-32 (x86-32)
- Backward compatible with old CPUs
- New mode: long mode (64-bit) – has compatibility sub-mode
- 64-bit memory addressing and common registers
- Doubled registers size: AX->EAX->RAX, DX->EDX->RDX,etc
- Single calling convention, stdcall and similar are ignored
- More strict requirements for data alignment, especially for stack

## Introduction - Modern platform limits

- Address space limitations
  - Theoretical maximum is 2^64 bytes = 16 Eb (Exabyte's) – as opposed to 4 Gb limit in Win32
  - Current implementations use only up to 48bits, which is equal to 2^48 = 256 Tb
  - The limit is expected to be raised in the future without mayor changes
- Hardware limitations
  - Current implementations use only up to 52 bits for physical memory – 4 Pb (Petabytes)
  - The limit is expected to be raised in the future without mayor changes
- OS limitations (Windows)
  - 8 Tb for user mode (64-bit apps) – can be lower due to OS edition
  - 8 Tb for kernel mode
  - The limit is expected to be raised in the future without mayor changes
- Why limit?
  - Currently there is no need for full 64-bits; so we can have enough space for the foreseen future without paying full cost of 64-bit address translation
  - Windows developers do not want to claim that Windows will work on any configuration which wasn't tested in the labs – and it's hard to find x86-64 system with 356 Tb of memory ☺
  - 8 Tb is still HUGE: allocate 1 Mb per second and you'll need 3 months to run out!

## 32-bit/64-bit – 32-bit code on 64-bit platform

- Kernel code must be 64-bit; old drivers will not work
- 32-bit software is run WoW64 compatibility layer
- Hardware compatibility mode, no emulation, full speed
- 32-bit software can run on 64-bit OS
- 64-bit software can't run on 32-bit OS
- 32-bit and 64-bit software are partially isolated – extra actions required to communicate cross-bitness

## 32-bit/64-bit – Points to consider

- All new hardware is 64-bit capable
- Most popular OS installations is 32-bit, but 64-bit OS installations will grow quickly, because 32-bit OS has memory limitations while new hardware has more memory standard installed
- Most common software is 32-bit
- 32-bit software seamlessly works on 64-bit system (with few exceptions)
- 64-bit application will not run on 32-bit system
- A 64-bit process can't load a 32-bit DLL
- A 32-bit process can't load a 64-bit DLL

## 32-bit/64-bit – Who needs 64-bit

- Your application may benefit from large address space and large memory usage
- Your code needs to interact with 64-bit software:
  - You write extensions for 64-bit Explorer, for 64-bit Office or for 64-biut Internet Explorer
  - You write plug-ins for any other 64-bit applications
  - You need to use 64-bit DDL's in your applications
- All your hardware and software are 64-bit, so no point in writing 32-bit code
- If you choose 64-bit: you still need 32-bit version of your software so it can be run on most computers which are still 32-bit; i.e. you most deploy TWO versions of your application

## 64-bit data models – data models overview

**64-bit data models**

| Data model | SmallInt/ Word | LongInt/ LongWord | Integer / Cardinal | Int64 / UInt64 | Pointer | An example of an operating system with this model |
|---|---|---|---|---|---|---|
| LLP64 | 16 | 32 | 32 | 64 | 64 | Microsoft Windows (X64/IA-64) |
| LP64 | 16 | 64 | 32 | 64 | 64 | Most Unix and Unix-like systems such asSolaris , Linux , and Mac OS X |
| ILP64 | 16 | 64 | 64 | 64 | 64 | HAL Computer Systems Solaris port on SPARC64 |
| SILP64 | 64 | 64 | 64 | 64 | 64 | Unicos |

- Data model controls how application's code is written
- Data model is software choice, per compiler, but data model of the host OS usually dominates
- LLP64 – data model to be used in Windows, FreePascal and Delphi:
  - Only pointers are increased to 64-bit
  - General integer types are still 32-bit

## 64-bit data models – Delphi models overview

- Classic data model: there are **generic** and **fundamental** integer data types
  - Integer and Cardinal are **generic types**, their sizes are expected to vary (not to be confused with generics – parameterized data types)
  - Byte/ShortInt, Word/SmallInt, LongInt/LongWord, Int64/UInt64 are **fundamental types** – their size never changes
- Modern data model:
  - Pointer is either 61-bit (Win16), 32-bit (Win32) or 64-bit (Win64)
  - SizeOf both Integer and Cardinal are 32 bits both for Win32 and Win64
  - Integer and Cardinal **are still generic types with variable size**. They provide best performance for the underlying CPU and operating system, but they **do not have to be the same size as Pointer**
  - Set of new (U)IntX data types which are aliases for fundamental data types
  - New data type: NativeInt/NativeUInt = a generic integer data type which size is always equal to SizeOf(Pointer) - **guarenteed**

## 64-bit data models – Why SizeOf(Integer) = 4

- Integer is still a general-purpose data type, which results in best performance - such type for X86-32 is 32-bit
- This is the same model as selected by OS
- Backward compatibility – too many code assumes it
- Most applications don't need 64-bits
- "Integer is 32-bit on hardware level"
  - X86-64 is not a real 64-bit solution as IA-64. It's just an extension to IA-32
  - Old 32-bit code can run full speed in 64-bit compatibility sub-mode, but this means 32-bit is default

## Moving your application to x86-64 - Rules

- If you need just generic integer type for common use – use Integer or Cardinal (just like always)
- If you need data type of a fixed size (typically as data exchange format) – use IntX or UIntX (X=8/16/32/64)

  Don't use Integer or Cardinal!
  - You can also use old names like SmallInt or ShortInt, but it's less preferable
- If you need to perform pointer calculations – use PByte (Delphi XE2) or PAnsiChar (Older Delphi versions)
  - If you're not satisfied with PByte/PAnsiChar or need to have integer representation of Pointer – use NativeUInt (or NativeInt)
  - The same goes for Pointer-like data (Numeric, Tag, handles, etc) - use NativeUInt (or NativeInt)

## Moving your application to x86-64 – Common issues

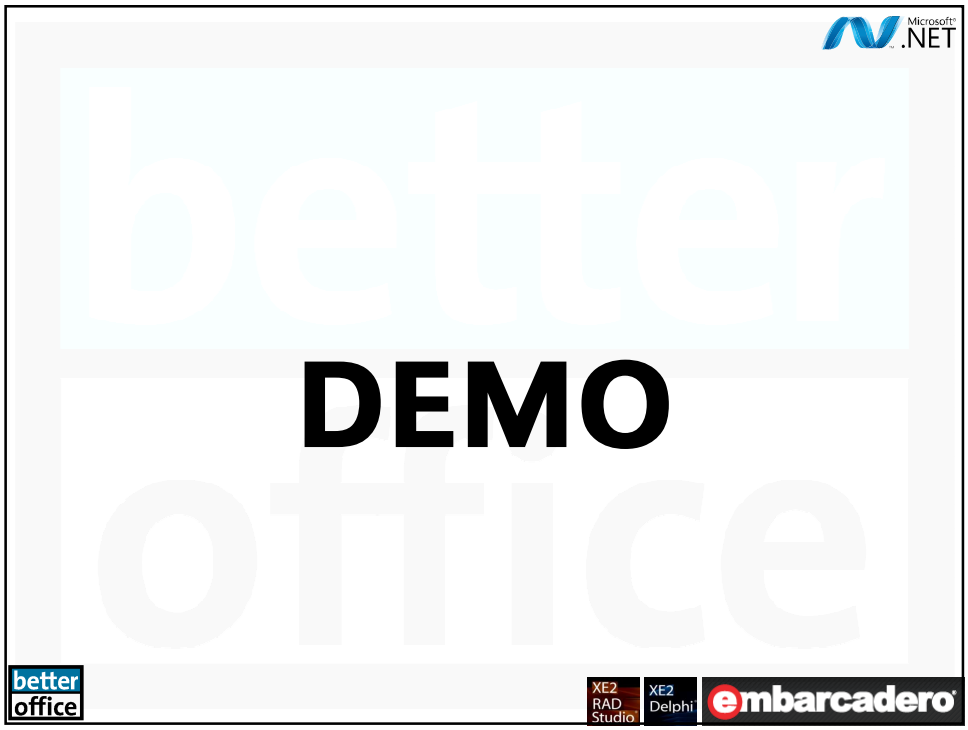
- **Biggest problem**: Assembly code, if you have assembly code in your code you have to rewrite it from scratch. Don't mix Assembly and Pascal code in one method
- **No** need to review your code for data exchange (files/MMF/etc)!
  - Old data types have the same size
- Review your code for pointer operations and casting Integer <-> Pointer
  - Replace Integer with NativeUInt
- Review your code
  - TComponent Tag property is 64-bit
  - LRESULT, WPARAM, and LPARAM are 64-bit. Ensure your message handling is adequate
  - Extended is 8 bytes (instead of 10) on Win64, so there is actually a loss of precision, compared to Win32
- Code doesn't compile? Function prototypes have changed
  - Got "types must be the same" error? Change type of variables passes as arguments to NativeInt or NativeUInt
  - Got "function from interface is not implemented by class" error? Check declaration of interface and copy it to your class
  - Got "function declaration differs" or "not declared in base class"? Check declarations?
- NativeInt exists on some old Delphi versions (Delphi 7 and older)
  - Ironically, it has SizeOf = 8 bytes (64 bit) – obviously wrong!
  - Use Navite(U)Int if you need to write code for Delphi 2007+ only
  - Otherwise define an analog of Native(U)Int – say, Ptr(U)Int – and use it instead

## Moving your application to x86-64 – Common issues

- Delphi bug, converted project file doesn't allow debugging when running as 64-bit application. Rebuild project file from scratch. Will be fixed in next update.
- Compiler Output Directory: .\$(Platform)\$(Config)
- Use your build configurations
- Define your target platforms

## References

- Converting 32-bit Delphi applications to 64-bit Windows
  http://docwiki.embarcadero.com/RADStudio/en/Converting_32-bit_Delphi_Applications_to_64-bit_Windows
- Delphi Considerations for Cross-Platform Applications
  http://docwiki.embarcadero.com/RADStudio/en/Delphi_Considerations_for_Cross-Platform_Applications
- 64-bit Cross-Platform Application Development for Windows
  http://docwiki.embarcadero.com/RADStudio/en/64-bit_Cross-Platform_Application_Development_for_Windows
- Programming Guide for 64-bit Windows
  http://msdn.microsoft.com/en-us/library/bb427430(VS.85).aspx
- AMD64 in FreePascal
  http://wiki.freepascal.org/Win64/AMD64_API
- Delphi converted project file 64-bit debugger bug filed in QC
  http://qc.embarcadero.com/wc/qcmain.aspx?d=100309


Microsoft .NET
DEMO
better office
XE2 RAD Studio
XE2 Delphi
embarcadero