

Microsoft®

# Tech·Ed 03

**10th Anniversary**

1011000101110010100101000**10**1010010110001011100101010

Microsoft®  
**Tech·Ed 03**

**10th Anniversary**

**DEV398**

**Porting Applications  
to Windows® for  
AMD64 Technology**

**Mike Wall**

**MTS Software Engineer, AMD**

1011000101110010100101000**10**1010010110001011100101010

# Agenda

- AMD64 technology
  - AMD64 Instruction Set Architecture
  - AMD Opteron™ and AMD Athlon™ 64 Processor overview
  - Platform features and multiprocessing
- 64-bit Windows® for AMD64
- What to port, and how
- Maximize multiprocessor performance
- Tools and additional resources

# Windows® and AMD64 Technology

## Unifying theme: Compatibility

- Processor: Native hardware support for 32-bit and 64-bit x86 code
- OS: 64-bit Windows® runs 32-bit and 64-bit applications side by side, seamlessly
- Code: A single C/C++ source code tree compiles to both 32-bit and 64-bit binaries

# AMD64 Technology

## AMD64 Instruction Set Architecture

# AMD64 Technology

## AMD64 Programmer's Model



# AMD64 Technology

## AMD64 Instruction Set Architecture

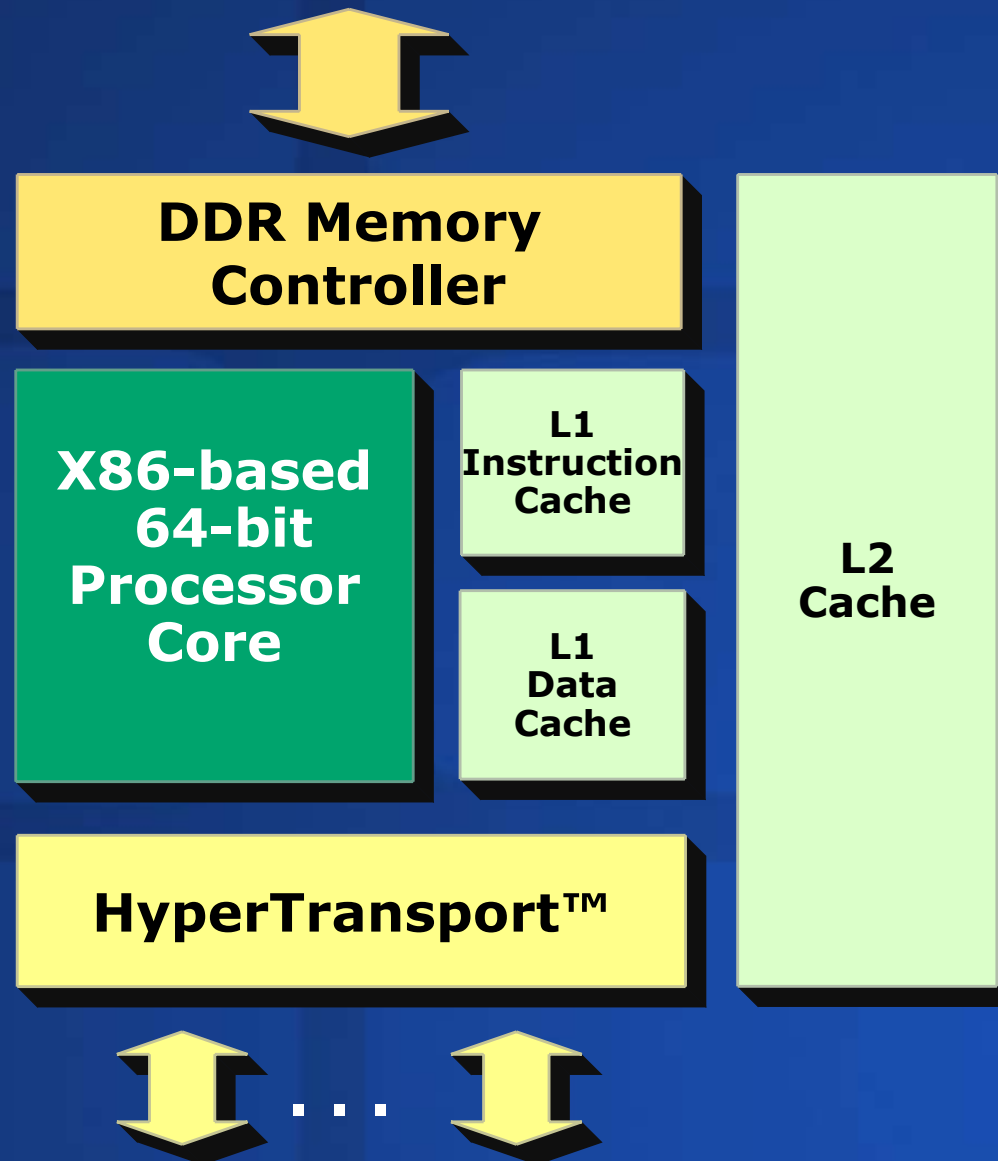
- Support for all x86 instruction extensions
  - MMX™, SSE, SSE2, 3DNow!™
- Full performance with all code
  - Native 32-bit x86 mode
  - Enhanced capability in 64-bit mode
    - 64-bit general purpose registers
    - 64-bit addressing
    - Twice as many general purpose registers
    - Twice as many SSE registers
- Same familiar x86 instructions

# AMD64 Technology

## AMD Opteron™ and AMD Athlon™ 64 Processor Overview

# AMD64 Technology

## AMD64 CPU block diagram



# AMD64 Technology

## Integrated memory controller

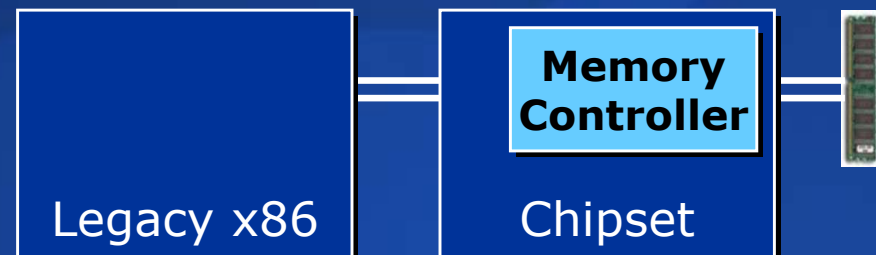
- The word to remember:

Latency

*1,000's of MHz  
& Always Increasing*



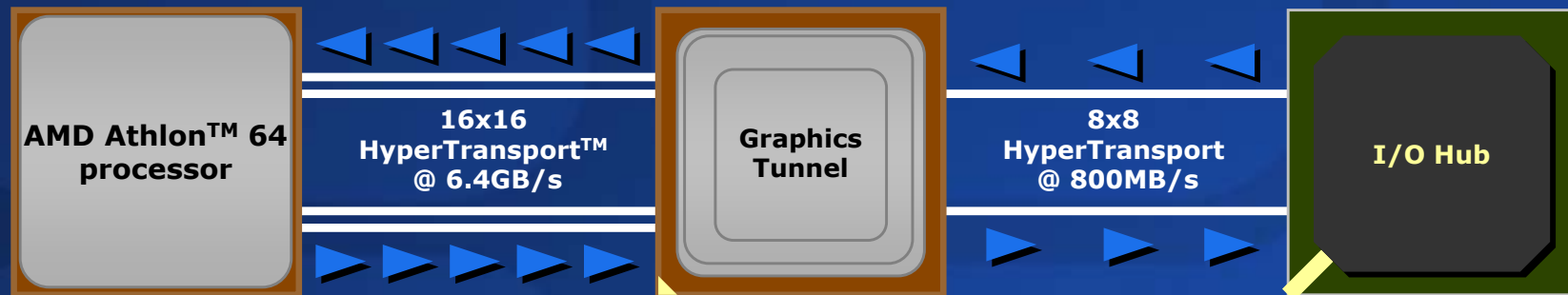
*100's of MHz  
& Not Improving*



- Integrated Memory Controller runs at CPU Core Frequency
  - As CPU frequency increases, the integrated memory controller becomes more efficient, but the Legacy x86 memory controller does not.

# AMD64 Technology

## HyperTransport™ Interface



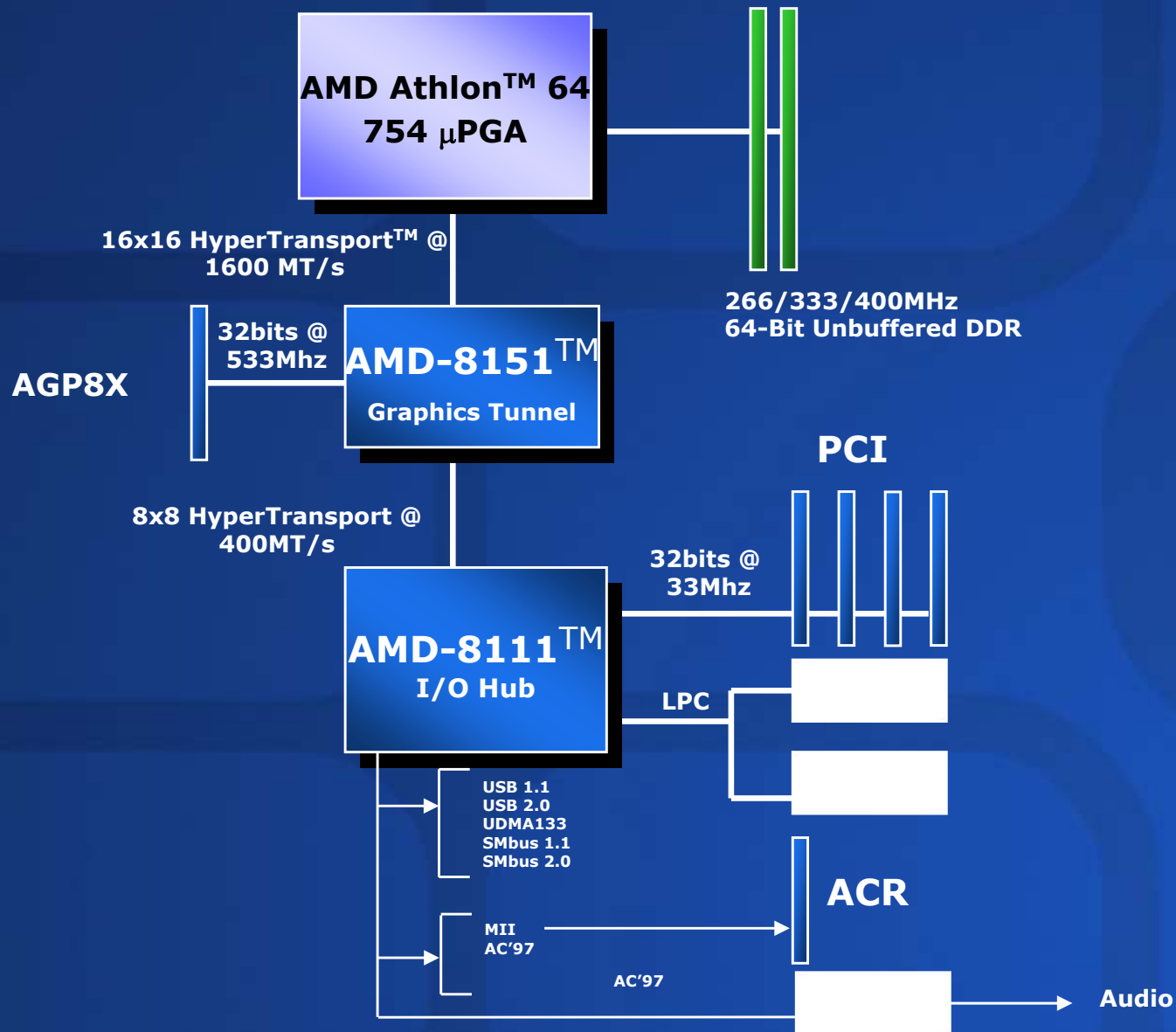
- **HyperTransport™ Technology Attributes**
- **Unidirectional (a pair of links)**
- **DDR-like performance (800MHz = 1600MT/sec)**
- **4 bytes wide ... 6.4GB/sec bandwidth**

# AMD64 Technology

**Platform features and multiprocessing**

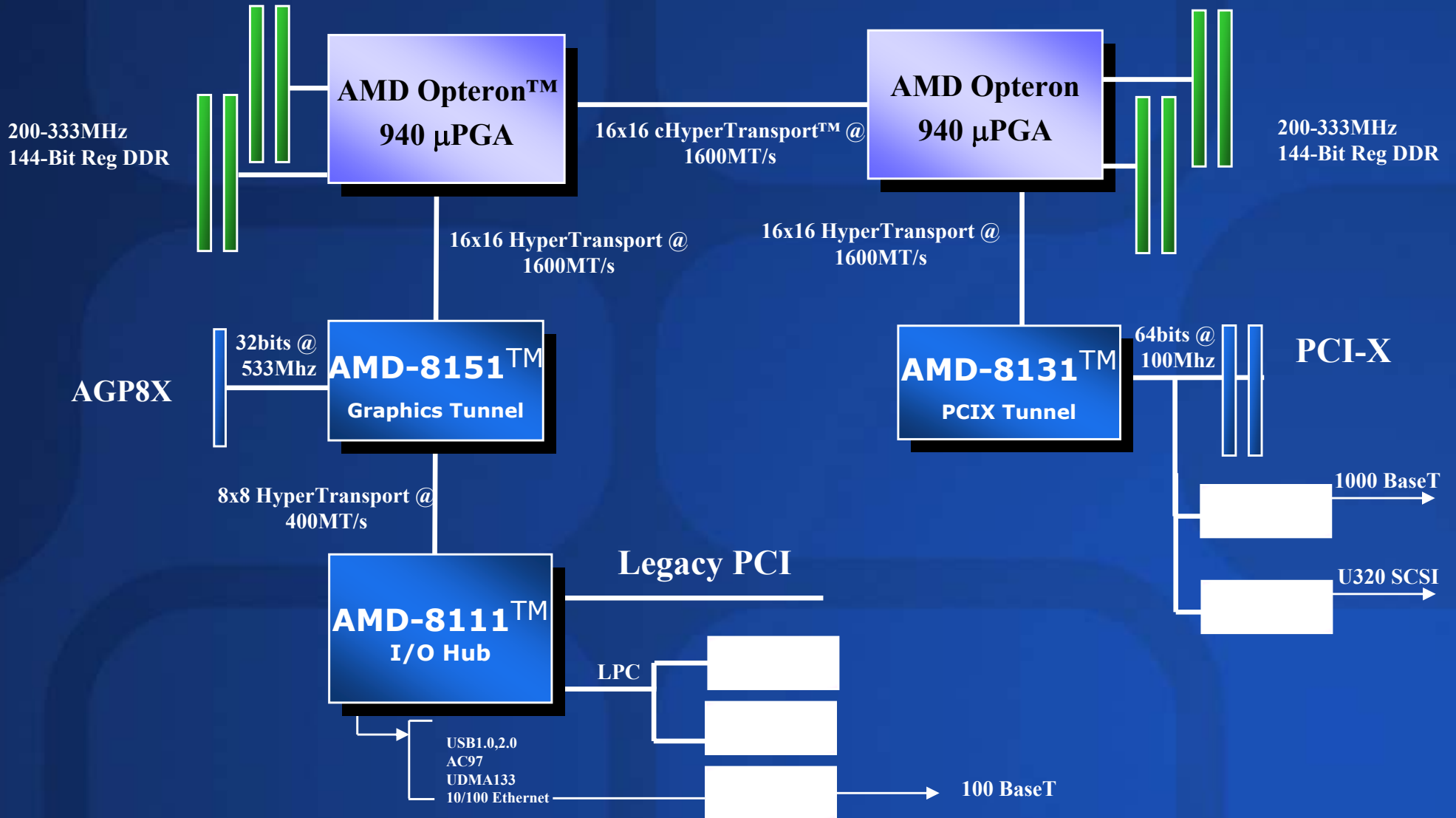
# AMD64 Technology

## Performance desktop PC



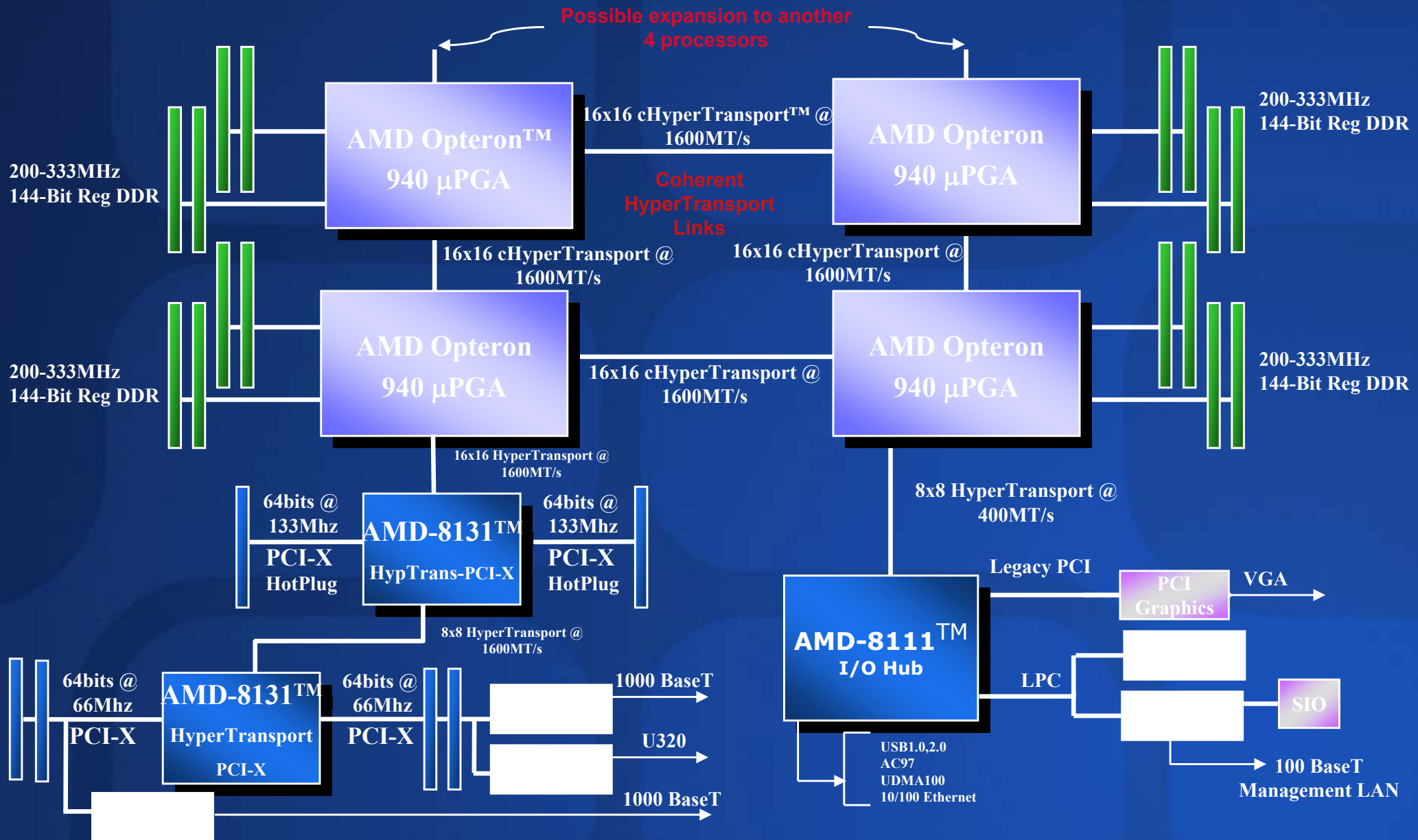
# AMD64 Technology

## High performance workstation



# AMD64 Technology

## 4P server

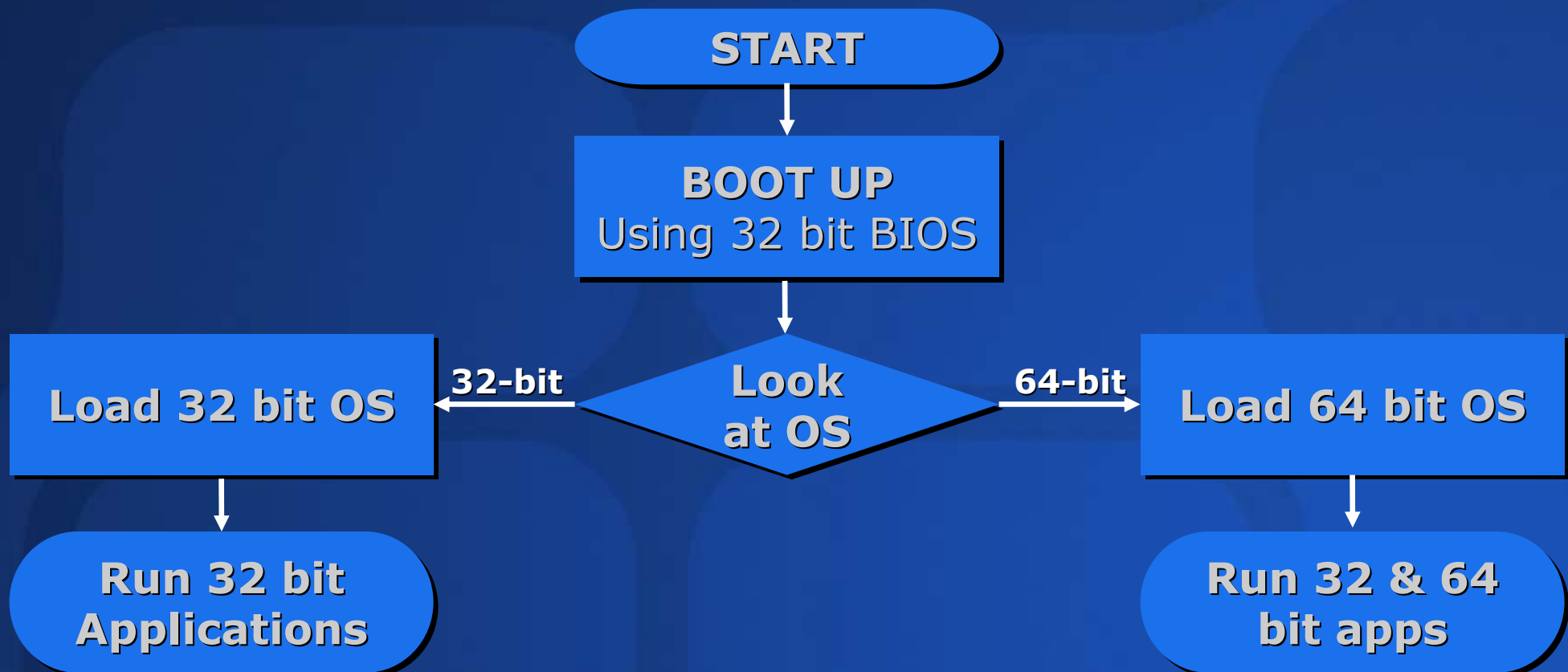


# **64-bit Windows® for AMD64 Technology**

# 64-bit Windows® for AMD64

## 32-bit and 64-bit on a single platform

- *An AMD64-based Processor can run both 32- and 64-bit Windows® operating systems*



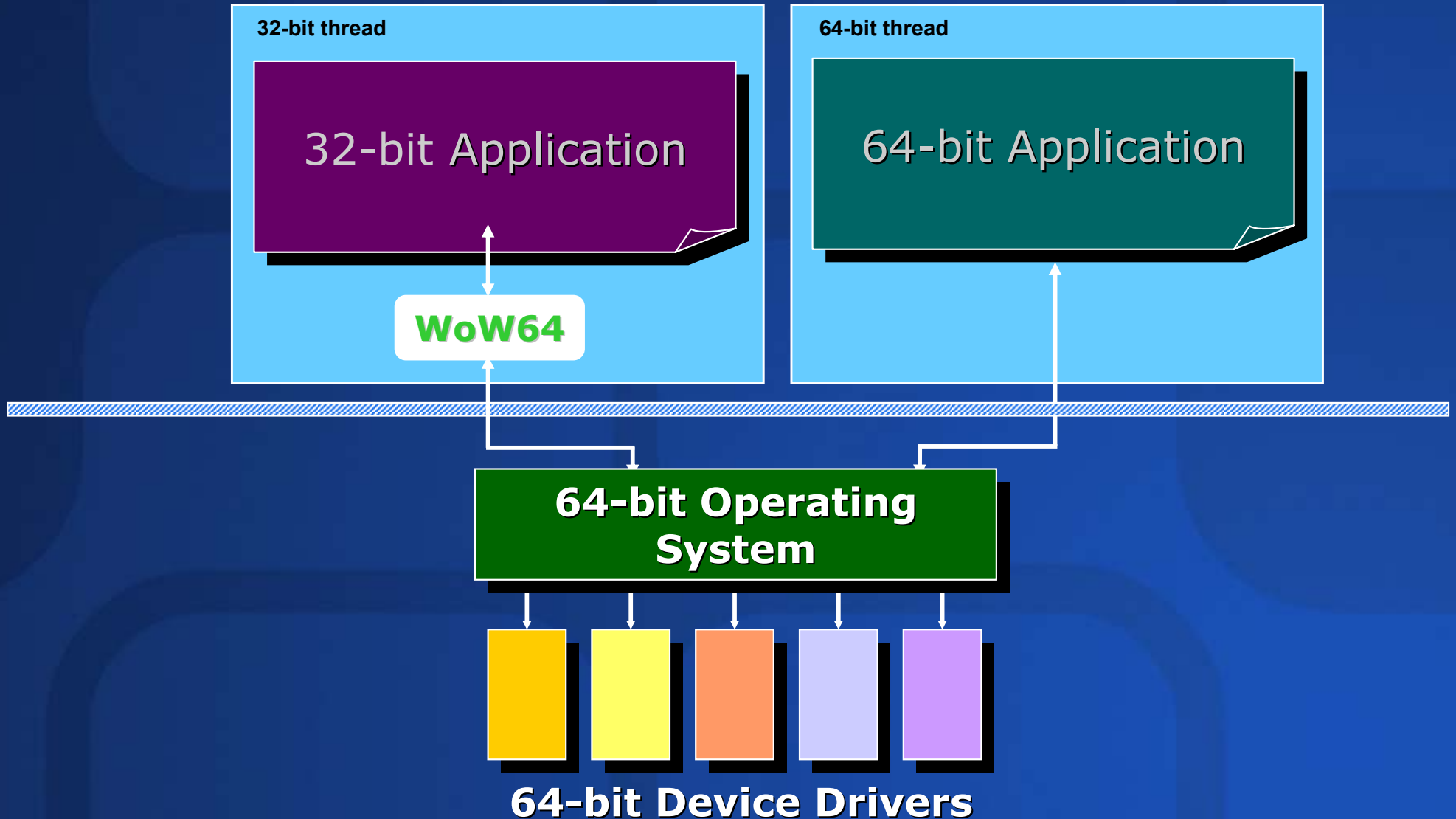
# 64-bit Windows® for AMD64

## Full backward compatibility with 32-bit

- Existing 32-bit applications run great
- WoW64 = Windows-on-Windows 64
- 32-bit applications run *on the hardware*
  - *no emulation anywhere with AMD64*
- Performance in WoW64 is not impaired
  - slight overhead for OS calls
  - sometimes compensated by faster OS
- Users can keep all their existing software

# 64-bit Windows® for AMD64 Technology

## WoW64 operation and compatibility



# AMD64: What to port, and how

## What apps will benefit from 64 bits?

- Large memory!
  - essentially unlimited virtual address space
  - physical memory only limited by platform
  - 8GB per CPU is expected to be common next year
- More registers and “big number” math
  - Codecs, simulation, 3D, games
  - Compression, encryption, finance
- Not everyone needs to port: 32-bit runs fine!

# AMD64: What to port, and how

## Some code really *must* be ported

- Drivers
  - device drivers must “match the OS”
  - 64-bit OS requires 64-bit drivers
  - there are presentations focused on drivers
- Code libraries and .dll's
  - customers who port will require 64-bit versions
- You can't mix 32 and 64-bit application code
  - but OS IPC mechanisms work 32  $\leftrightarrow$  64

# AMD64: What to port, and how

## Moving up to 64-bit mode

- A *single* source code tree → compile to 32-bit *and* 64-bit Windows® platforms!
- Programming for 64-bit Windows® is basically the same as 32-bit
  - same API, a few different data types
- Types `int` and `long` remain 32 bits
- Only pointers expand to 64 bits “P64”
- Use `size_t` and new polymorphic types

# AMD64: What to port, and how

## Use portable / scalable data types

- Use `int` where only 32 bits are needed
  - e.g. Index variable for a small array
- Use `size_t` where array may grow beyond the 32-bit limit
- Special *polymorphic* types for pointer math
  - `INT_PTR`, `UINT_PTR`, `LONG_PTR`,  
`ULONG_PTR`
  - These types scale to match the 32/64-bit mode

# AMD64: What to port, and how

## Code example #1

- `char *p;`
- `long Lval;`
- `Lval = p; // bad! truncated in 64-bit mode`
  
- `char *p;`
- `LONG_PTR Lval; // either 32 bits or 64 bits`
- `Lval = p; // works fine`

# AMD64: What to port, and how

## Code example #2

```
for (int i = 0; i < S; i++) {  
    A[i] += 3;  
}
```

If S may exceed 32 bits, use:

```
for (size_t i = 0; i < S; i++) {  
    A[i] += 3;  
}
```

# AMD64: What to port, and how

## Code example #3

- objectStart = (PVOID)  
( ( **ULONG** ) objectStart | 1 );
- This is bad because we are casting a pointer to a long; it will get truncated when we compile for 64-bit
- objectStart = (PVOID)  
( ( **ULONG\_PTR** ) objectStart | 1 );
- Proper use of polymorphic data type

# AMD64: What to port, and how

## Assorted portability tips

- `-1 != 0xFFFFFFFF` (just use “-1”)
- Many Windows® apps use `DWORD`
  - a `DWORD` is always 32 bits, make sure that is really what you want
- Use `%p` in a `printf`, to print a pointer
- Data alignment can affect performance
  - there are alignment requirements in the ABI
  - structure padding can affect portability

# AMD64: What to port, and how

## Explicitly sized types

- `INT64` and `UINT64` are always 64 bits
- `INT32` and `UINT32` are always 32 bits
- Think carefully before using these
  - most often you really want `INT_PTR` etc.
  - explicit size type are useful for shared data
    - but you can typically just use `int` and `long`

# AMD64: What to port, and how

## Microsoft C compiler and libraries

- The 64-bit compiler emits SSE / SSE2 code
- The string and memory functions are optimized and should be used
- AMD provides optimized math libraries:
  - ACML = AMD Core Math Libraries (BLAS, FFT, LAPACK, and more)

# AMD64: What to port, and how

## Microsoft C compiler and libraries (2)

- In-line assembly code is not supported
  - put assembly code in a separate MASM file
- Compiler intrinsics are provided for effectively in-lining many SSE functions and other special functions

# AMD64: What to port, and how

## Porting x86 assembly code

- Assembly code is straightforward to port
  - And worth doing for performance reasons
- In-line assembly no longer supported
  - Use separate MASM file, compile .obj and link
- Take advantage of the new registers!
  - GPR regs r8-r15, SSE regs xmm8-xmm15
- New 64-bit stack frames, calling convention
  - Carefully read the ABI document!
    - →→ “AMD64 Software Conventions”

# AMD64: What to port, and how

## Using the 64-bit registers

32-bit

- `mov edx, 66`
- `mov eax, [ecx + edx*4]`
- `mov ecx, [eax] //32-bit pointer`

64-bit

- `mov r10, 66`
- `mov rax, [rcx + r10*8]`
- `mov rcx, [rax] //64-bit pointer`

# AMD64: What to port, and how

## 64-bit regs: upper half gets cleared

32-bit code might look like this

- `xor edx, edx // clear all bits`
- `mov dx, 66 // load 16-bit value`

64-bit code looks like this

- `mov edx, 66 // load 32-bit value`
- `mov ecx, 0 // upper half is cleared`

# AMD64: What to port, and how

## Function call args passed in regs

- 32-bit code might look like this

- `push eax // use stack`
  - `push ecx`
  - `call func`

- 64-bit code looks like this

- `mov rcx, n // up to 4 args in regs:`
  - `mov rdx, m // can use rcx,rdx,r8,r9`
  - `call func`

# AMD64: What to port, and how

## Only SSE and SSE2 for 64-bit code

- 64-bit native applications use SSE/SSE2
  - single precision floating point
  - double precision floating point
  - integer SSE2 vector operations
- Legacy 3DNow!<sup>™</sup>, x87 and MMX<sup>™</sup> are not supported for native 64-bit apps
  - still fully supported for 32-bit apps on 64-bit Windows<sup>®</sup>
- SSE / SSE2 is the way forward

# AMD64: What to port, and how

## Convert MMX™ code to integer SSE2

- The instructions are the same
  - but the registers are 128 bits instead of 64
  - take care to align your data and use MOVDQA
    - QWORD (16-byte) aligned MOVs are faster for SSE

### MMX™ instructions

PADDB	MM0, MM5
PMULLW	MM3, [EAX+8]
PCMPEQB	MM4, MM7



### SSE2 instructions

PADDB	XMM0, XMM5
PMULLW	XMM3, [RAX+16]
PCMPEQB	XMM13, XMM12

# AMD64: What to port, and how

## Convert x87 code to SSE / SSE2

- The compiler generates SSE2 for floating point data types, both single and double precision
- Assembly code x87 must be manually converted
- Flat addressing of SSE registers is better
- 128-bit SSE register size supports vectorization

# AMD64: What to port, and how

## Take advantage of 64-bit address space

- MapViewOfFile and CreateFileMapping
  - Maps a file as a chunk of memory
  - Access type can be read, write, copy
  - In 64-bit mode this can be used with large files
  - Let Windows® manage your physical memory
- This approach is especially interesting with 64-bit addressing: “unlimited” virtual space
- Simplify your application programming model

# AMD64: What to port, and how

## Take advantage of “big number math”

- Special compiler intrinsic functions
  - CPU ID, SSE, SSE2
- Intrinsic functions for AMD64
  - `LONG64 __mulh ( LONG64, LONG64)`
    - Returns the high 64 bits of a 64x64 multiply
  - `ULONG64 __rdtsc(VOID)`
    - Read time stamp counter, great for benchmarking
- Many more! Go to MSDN and search for “AMD64 intrinsic functions”

# AMD64: What to port, and how

## First steps to prepare for porting

- Compile 32-bit code using /Wp64 switch
  - warn about portability issues
  - get your code “64-bit clean”
- Take inventory of code dependencies
  - examine your project files, makefiles, etc.
  - .lib and .dll files all must go 64-bit too

# AMD64: What to port, and how

## Go ahead and port it 8-)

- Compile with 64-bit compiler and tools
  - you may find some additional bugs
  - be careful to keep sync between C and ASM data structures, new data sizes/alignments
  - if your application uses shared files or networked data, you may need to convert inside your 64-bit app
- Benchmark and tune for best performance
  - tools and documents are available

# AMD64: What to port, and how

## DirectX and DirectShow

- 32-bit DirectX applications supported
  - WoW64 runs existing DirectX apps
- Microsoft always recommends using the latest released version of DirectX for titles under development
  - DX9 has been the current version since January 2003
- Early adopters: ask Microsoft for guidance
  - E-mail Microsoft's AMD64 Support at [x64info@microsoft.com](mailto:x64info@microsoft.com)

# 64-bit Windows® for AMD64

## Multiprocessor performance

- Consider the NUMA platform
  - some physical RAM is local, some isn't
  - local RAM is somewhat faster (not a lot)
- Windows® implements ccNUMA support
  - malloc returns local memory when possible
  - allocate from the right thread/process
  - use ccNUMA API funcs if you need more control over thread or process assignment
- Performance benefits on both 32 and 64-bit

*demo*

**32 and 64-bit demo**

***announcing. . .***

**Hands-On Porting Lab  
at the AMD exhibit booth**

# Ask The Experts

## Get Your Questions Answered

- Available at the Ask the Experts area today (Thursday) 2:00PM - 4:00PM
- You're welcome to ask questions or just stop by and say hello
- I am always interested to hear about your applications, adventures in programming, software optimization or whatever

# Community Resources

- Community Resources  
<http://www.microsoft.com/communities/default.mspx>
- Most Valuable Professional (MVP)  
<http://www.mvp.support.microsoft.com/>
- Newsgroups  
Converse online with Microsoft Newsgroups, including Worldwide  
<http://www.microsoft.com/communities/newsgroups/default.mspx>
- User Groups  
Meet and learn with your peers  
<http://www.microsoft.com/communities/usergroups/default.mspx>

# More Resources - AMD

- AMD Developer Center
  - <http://www.developwithamd.com/devcenter>
  - come to California, use our machines, talk with our tech people
  - remote access via VPN
- AMD tools and documentation
  - <http://developer.amd.com>
  - CodeAnalyst profiler, Optimization Guide, Programmer's Manuals, other tech docs.
  - AMD64 Developer Resource Kit

# More Resources - Microsoft

- Search MSDN for “AMD64” and “64-bit”
- Join the Microsoft Windows® for AMD64 Beta Program if you plan to produce products supporting this platform
  - e-mail [x64info@microsoft.com](mailto:x64info@microsoft.com)

***evaluations...***

**Don't forget to  
complete your online  
Evaluation Form!**

AMD, the AMD Arrow Logo, AMD Athlon, AMD Opteron, and combinations thereof, 3DNow!, AMD-8111, AMD-8131, and AMD-8151 are trademarks of Advanced Micro Devices Inc. HyperTransport is a licensed trademark of the HyperTransport Technology Consortium. Windows is a registered trademark of Microsoft Corporation. MMX is a trademark of Intel Corporation. Other product names used in this publication are for identification purposes only and may be trademarks of their respective companies.

***Microsoft***<sup>®</sup>