عنوان: **آیا برنامه نویس‌های دات نت باید نگران دنیای 64 بیتی باشند؟**
نویسنده: وحید نصیری
تاریخ: ۱۳۸۷/۱۱/۳۰ ۱۸:۲۱:۱۲
آدرس: www.dotnettips.info
برچسب‌ها: Tips

جواب ساده و کوتاه: خیر!
کدمدیریت شده‌ی شما در هر دو پلتفرم 32 بیتی - x86 و x64 بدون نیاز به هیچگونه تغییری و بدون نگرانی اجرا خواهد شد.
گزیده‌ای از MSDN :
اگر کد شما 100 درصد مدیریت شده است (managed code ایی که به صورت خالص از دات نت فریم ورک استفاده می‌کند و هیچگونه وابستگی خارجی دیگری به کتابخانه‌های دیگر ندارد)، تنها با کپی شدن در یک محیط x64 دارای CLR ایی 64 بیتی (دات نت فریم ورک 64 بیتی)، بدون هیچگونه مشکلی اجرا خواهد شد.

سؤال: چرا و چگونه؟!
کامپایلرهای دات نتی (تفاوتی نمی‌کند که چه زبانی مورد استفاده باشد)، کد شما را به IL ترجمه می‌کنند و IL اساسا درکی از پروسسور ندارد. JIT است که در آخرین لحظه در این مورد تصمیم گیری می‌کند.

این نگرانی از کجا حاصل شده است؟
نگارش R2 ویندوز 2008 سرور، فقط 64 بیتی خواهد بود و ویندوز سرور 2008 فعلی، آخرین سروری از مایکروسافت است که هر دو نسخه‌ی 32 بیتی و 64 بیتی را دارد. بنابراین دیر یا زود تمام برنامه نویس‌های ویندوزی "مجبور" خواهند شد دنیای 64 بیتی را تجربه کنند. (البته اگر تاکنون آن‌را تجربه نکرده‌اند)
و البته هنوز یک سری از محیط‌های توسعه، کامپایلر مخصوص 64 بیتی ندارند (مانند دلفی که قرار است در طول سال جاری اولین تجربه‌ی 64 بیتی خود را ارائه دهد)

نکته:
در صفحه‌ی build ویژوال استودیو، شما می‌توانید نوع پلتفرم مورد نظر را نیز تعیین کنید:

پیش فرض آن بر روی Any CPU است و در این حالت کد کامپایل شده‌ی شما بدون مشکل بر روی پلتفرم‌هایی که مشاهده می‌کنید اجرا خواهد شد و تنها پیشنیاز اجرای آن، نصب نسخه‌ی دات نت فریم ورک مخصوص آن پلتفرم است، بدون اینکه نیاز باشد برنامه نویس نگران جزئیات خاصی در مورد خصوصیات ویژه‌ی آن پلتفرم ویژه باشد.

سؤال: اگر کد ما خالص نبود چطور؟ (منظور اینکه 100 درصد دات نتی نبود)

حالت الف) اگر از کامپوننت‌های خارجی استفاده می‌کنید (حتی اگر 100 درصد دات نتی هم باشند) حتما اطمینان حاصل کنید که برای پلتفرم خاصی کامپایل نشده‌اند (همان Any CPU مورد استفاده بوده)، زیرا کد شما که برای تمام CPU ها کامپایل شده، در محیط 64 بیتی، تنها توانایی بارگذاری اسمبلی‌های 64 بیتی را خواهد داشت (64 بیتی رفتار می‌کند) و با مواجه شدن با اسمبلی‌هایی که برای یک پروسسور خاص دیگر کامپایل شده‌اند، با خطای BadImageFormatException خاتمه می‌یابد.

حالت ب) استفاده از API ویندوز یا DLL های غیر دات نتی
باید با هماهنگی با تولید کننده‌ی مربوطه حتما از نگارش 64 بیتی استفاده شود و همچنین برنامه‌ی شما باید توانایی استفاده از اشاره‌گرهای 64 بیتی را داشته باشد. اندازه‌ی نوع داده‌ای IntPtr در یک محیط 32 بیتی 4 است و در یک محیط 64 بیتی 8 خواهد بود (IntPtr.Size). اگر در حین اجرای ترجمه‌ی API یک کتابخانه به اشتباه بجای استفاده از IntPtr از int استفاده شده باشد، ممکن است کد شما در یک محیط 32 بیتی سال‌ها بدون مشکل اجرا شود، اما در اولین اجرای خود در یک محیط 64 بیتی، کرش خواهد کرد. (بدلیل overflow حاصل)

IntPtr به اندازه‌ی کافی هوشمند است تا سایز خودش را مطابق پلتفرم تنظیم کند و مشکل ساز نشود.

مثال:

```
[DllImport("kernel32.dll")]
public static extern void GetSystemInfo([MarshalAs(UnmanagedType.Struct)] ref SYSTEM_INFO
lpSystemInfo);

[StructLayout(LayoutKind.Sequential)]
public struct SYSTEM_INFO
{
 internal _PROCESSOR_INFO_UNION uProcessorInfo;
 public uint dwPageSize;
 public IntPtr lpMinimumApplicationAddress;
 public int lpMaximumApplicationAddress;
 public IntPtr dwActiveProcessorMask;
 public uint dwNumberOfProcessors;
 public uint dwProcessorType;
 public uint dwAllocationGranularity;
 public ushort dwProcessorLevel;
 public ushort dwProcessorRevision;
}

[StructLayout(LayoutKind.Explicit)]
public struct _PROCESSOR_INFO_UNION
{
 [FieldOffset(0)]
 internal uint dwOemId;
 [FieldOffset(0)]
 internal ushort wProcessorArchitecture;
 [FieldOffset(2)]
 internal ushort wReserved;
}
```

در این مثال که از API ویندوز استفاده می‌شود، به اشتباه نوع lpMaximumApplicationAddress به صورت int تعریف شده است (بجای IntPtr). این کد بدون مشکل در یک برنامه‌ی 32 بیتی کار می‌کند اما همین برنامه در یک محیط 64 بیتی یا کرش خواهد کرد یا مقدار lpMaximumApplicationAddress منفی گزارش می‌شود.