嵌入式系统大作业

学生姓名:

学　号:

专　业:

班　级:

选题　：嵌入式系统中操作系统
　　的框架结构分析

指导教师:

完成日期: 2019 年 12 月 6 日

目 录

1. Window CE 简介

1.1 Windows CE 来源

Windows CE 是微软公司为移动应用产品，消费电子产品和嵌入式应用产品等非 PC 领域产品设计的操作系统。

1.2 Windows CE 的主要功能

Windows CE 的特点是：可延伸性，实时性好，通信能力强，支持多种体系 CPU 架构。

从操作系统内核角度看，Windows CE 具有灵活的电源管理功能，同时使用了对象存储技术，如文件系统，注册表以及数据库。它还具有很多高性能，高效率的操作系统特性，包括视需要分页，共享存储，交叉处理以及支持大容量 heap 等。

Windows CE 具有良好的通信能力。它广泛支持各种通信硬件，支持直接的 LAN 连接以及拨号连接，且提供与 PC，Intranet 以及 Internet 的连接。

Windows CE 的界面功能相当出色，拥有基于 Microsoft IE 的 Internet 浏览器，同时支持诸如手写，声音，动态影像和 3D 图像等多种应用。

Windows CE 是个多任务操作系统，同时执行多个任务，并且在他们之间来回切换。

1.3 Windows CE 的实时性能

Windows CE 的实时能力可以适应 95%的硬件实时(hard-real-time)系统的需求。同时，由于中断延时也是影响实时性的主要因素，在 Windows CE 中，中断延时和中断处理方式密切相关，当采用在 ISR 中直接处理时，延时非常短，较长的延时通常发生在采用 IST 方式处理中断的情况下，进程系统保证此种情况下延时不超过 100 us。

为了保证实时性，Windows CE 在以下功能上有着卓越的表现：

- Bounder interrupt response latency(有限的中断响应)
- Timer precision configurable(系统时钟精度可配置)
- System time tick independent of thread quantum(时钟滴答不依赖线程时间片)
- Nested inheritance(支持中断嵌套)
- Priority inheritance(支持优先级继承)
- Protected virtual memory(受保护的虚拟内存)
- Synchronization objects(对象同步)

2. 嵌入式操作系统 Windows CE 的组建分析

2.1 Windows CE 的层次式架构

操作系统一般都具有分层的结构特征，典型的比如 Linux 操作系统的用同心圆结构来表示系统的用户空间和内核空间，其中内层表示内核空间，外层表示用户空间。Windows CE 的分层结构比较复杂，这是根据它适合嵌入式开发的特点所设计的。如图 2.1 所示，图中最上层由应用程序开发者开发，中间层由微软公司开发。最下层由嵌入式产品开发商根据自己的需求进行的硬件层以及外围电路的设计和开发。

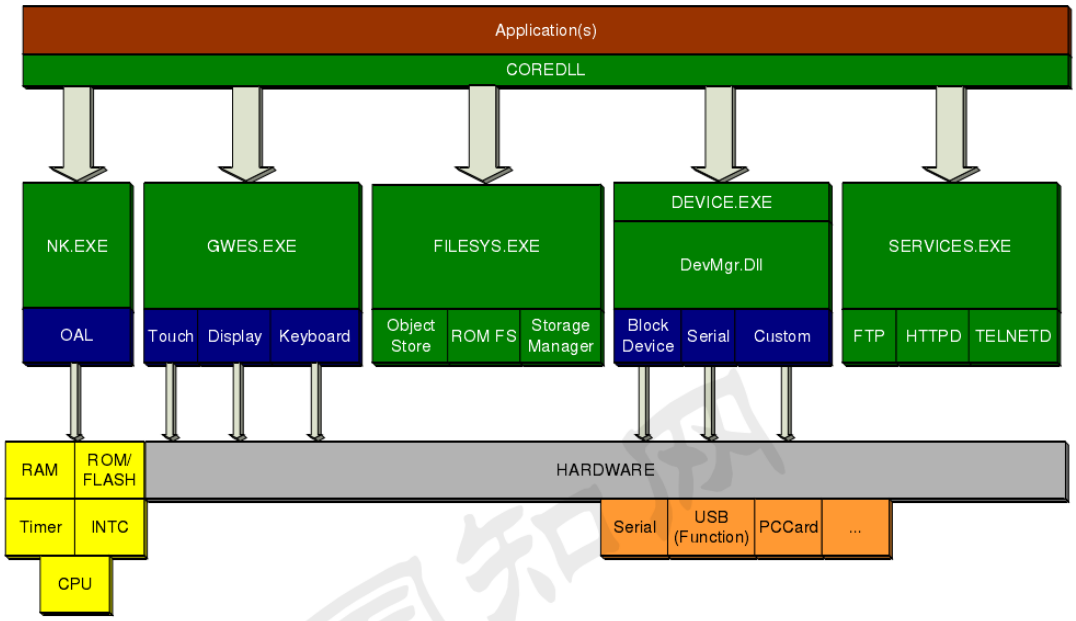

图 2.1 Windows CE 的分层结构示意图

从接口角度看，一般系统开发环境包括系统和应用两个接口，分别用以支持系统开发和应用开发，Windows 中 SDK 代表了应用层接口，DDK 代表了系统层接口，这两个接口之间的部分就是操作系统的实体。

2.2 Windows CE 的组件剖析

如上图 3.1 中绿颜色部分所示，它代表了 Windows CE 中最主要的几个组件或者模块，从宏观角度划分，系统的软件组件有 CoreDll，NK，设备管理，数据存储，GWES，通信，OAL 以及驱动和 Win32 系统服务模块。

- NK

NK 通过 Nk.exe 在系统中运行，他是 Windows CE 的真正核心，主要包含以下 6 类功能：功能处理器进程，内存管理，异常处理，系统内的通信机制，为其他部分提供核心应用程序 Routine，为系统内侦错提供支持。

- CoreDll

CoreDll 在系统中地位特殊，它把应用程序和操作系统分割成了两部分，是使系统保持稳定的保护性屏障。它提供两类功能：第一类是外部程序的系统功能的代理；第二类是提供了类似字符串处理，随机数生成，时间计算等基本的函数支持。

- 设备管理模块

Windows CE 操作系统的设备管理核心，通过 Device.exe 运行，负责提供系统范围内基本的设备列表管理，随插即用管理，电源管理，I/O 管理，并提供了设备驱动程序运行的基本机制。

- 数据存储模块

该模块主要是提供系统基本的数据存储能力，其中包括对象存储和文件系统，这些功能主要是通过 Filesys.exe 来执行。

- GWES 模块

Windows CE 通过该模块提供的图形接口提供以下几个共更能：基本的绘图引擎，窗口管理，接口事件机制等，运行时在系统中为 Gwes.exe。

- 通信模块

整个 Windows CE 模块中最为独立的部分，通过一系列的动态链接库来运作，涉及众多的 Windows 平台公用特性。

- 驱动程序模块

这个模块并不是一个独立的实体，而是一个由驱动程序实体构成的集合，包括很多组件，之行也比较复杂，实际上是多个其他模块的底层部分，如图 3.1 所表示。

- OAL 模块

这个模块没有固定的形态，主要是包括和硬件相关的若干功能。一般的，OAL 不具有可移植性。Win32 系统服务模块 Win32 系统服务是 Windows CE 对应用程序提供的接口。

3. 嵌入式操作系统 Windows CE 的内核模块

如上节所述，Windows CE 内核的功能主要由各个模块来完成，但是这些模块都不是完全独立的，因此，为了简单叙述和基于项目的需求，我们只对那些与项目需求联系紧密，功能作用强大的模块作一简单介绍。

3.1 Windows CE 的中断体系结构

Windows CE 的中断体系由硬件中断，内核中断服务例程，OAL 中断服务例程和中断服务线程 IST 组成。各个中断服务模型就是围绕着这 5 部分展开的。

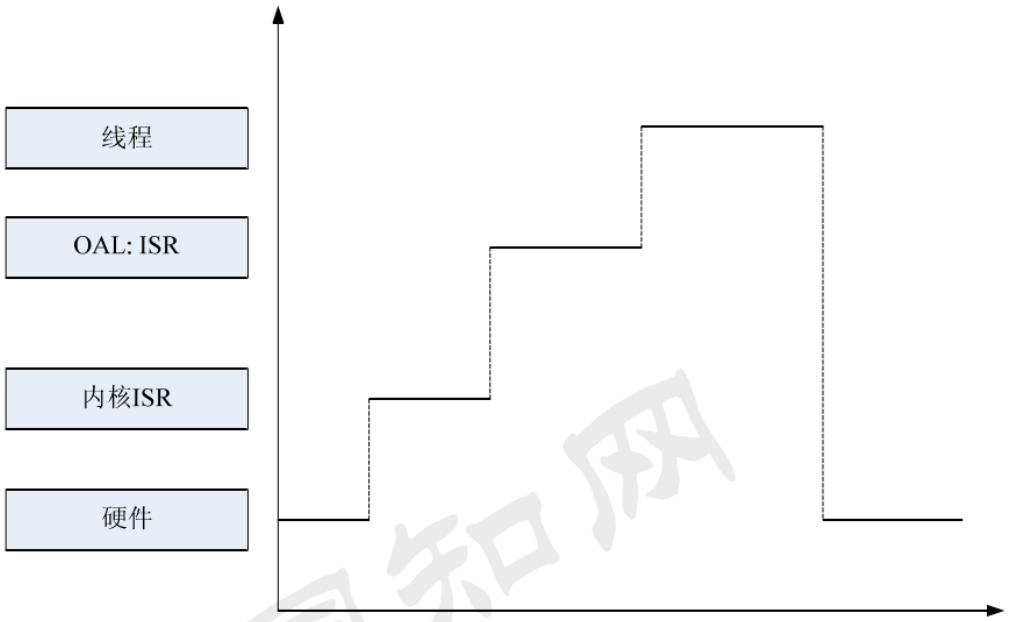

图 3.1 Windows CE 的中断体系结构

图 2.2 描述了中断过程的主要转换过程，纵轴代表抽象层次，横轴代表中断的发生以及处理时间顺序。

系统最底层是硬件以及中断控制器。外围设备通过向处理器的中断信号线上发送中断信号，就产生了中断。

第二层是内核 ISR。外设产生中断后，处理器立即转入中断服务向量表，即操作系统的中断服务向量表，然后搜索并执行中断服务程序；由于外设的中断可能不止有一个，可能会有多个中断而且具有多个优先级，因此，内核中断服务程序主要做的就是禁止处理器上的所有具有相同和较低优先级的中断在本中断处理过程中发生中断。

第三层是 OAL 的 ISR 例程。这层通过外设的生产商通过驱动实现设备的特定功能，往往这部分程序集成在系统的板载支持包 BSP 中。

最上层是驱动程序或者应用程序的中断服务线程 IST。在 OALISR 对中断

进行了简单处理之后，IST 就要具体的处理中断了。

3.2 Windows CE 的进程

在 Windows CE 中，进程是由正文段（text），用户数据段（user segment）以及系统数据段（system segment）共同组成的一个执行环境，负责处理器，内存以及外围等资源的分配和回收。进程是计算机系统资源的使用主体，是操作系统分配资源的基本单位。动态，独立，并行的 Windows CE 最多支持 32 个进程同时在系统中运行。

系统激活的时候，至少自动激活四个进程：第一个是 nk.exe，负责提供操作系统中的 kernal 服务；第二个是 filesys.exe，用来提供文件系统服务；第三个是 gwes.exe，用来提供对系统 gui 的支持；第四个是 device.exe，用来加载和管理外围的驱动程序。他们占据虚拟地址的四个 slot，每个 slot 有 32 个空间。

3.2.1 Windows CE 进程简介

作为一个抢占式，多任务的操作系统，Windows CE 在系统之内最高支持 32 个正在运行的进程。每个进程又由若干线程组成，其中有一个主线程，每个线程代表一个进程中一个独立的部分。

从 Windows CE 3.0 开始，系统提供了 256 个优先级，0 为最高，255 为最低的优先级。他们的优先级分配在 Windows CE 中大致如下表 3.1 所示。

表 3.1 Windows CE **的优先级分组**

Levels	Description
0～90	为实时性要求比驱动强的实时程序保留
97～152	为基本的 Windows CE 驱动程序保留
153～247	为实时性要求比驱动弱的实时程序保留
248～255	为非实时程序保留

3.2.2 Windows CE 进程间通信

Windows CE 处理程序间的通信（IPC，Inter Process Communication）可以

使用以下几种方法：临界区（critical section），事件（event），同步（synchronization）以及互斥器（mutex）。

① 临界区（critical section）

当多个线程同时得到同一对象的访问权后，他们对对象的操作可能导致数据的不一致。使用临界区对象，可以使得某个对象或者某段代码避免被多个线程同时修改。临界区使得对象或者代码段只能被一个线程来操作，他们可以是一个 DLL 或者一个进程空间中的线程。这是系统内部最基本的互斥方式，通常用于重要资料的修改。一般情况下，临界区对象用于保证一段程序代码的原子性。临界区对象的使用被限制在某一个处理程序的上下文，不能被多个进程同时使用。下面这段代码样例可以显示如何在一个线程中初始化，进入，离开临界区，代码使用了 try-catch 异常捕获机制。

```
void test( )
{
    //获取进入临界区的权限
    EnterCtriticalSection(&sampleCriticalSection);
    //这将使得进程阻塞，直到他能获得这样的权限
    try
    {
   //这里实现对资源的操作
    }
    catch(EXCEPTION e)
    {
    //异常处理
    }

    //解除临界区进入权限
    LeaveCriticalSection(&sampleCriticalSection);
}
```

使用 Initialize Critical Section 可以向系统取得一个临界区对象，使用完

毕之后，需要调用 Delete Critical Section 释放该资源。一段临界代码执行时，相关的临界区对象句柄必须有效，这段程序代码需要以 Enter Critical Section 或者 Try Enter Critical Section 函数开始，以 Leave Critical Section 结束。这样，保证多个进程试图同时进入同一段临界程序代码的时候，只有一个能获得进入权限。

② 事件（Event）

事件对象用以通知某个进程发生了特定的时间或者告诉这个进程该做什么事情了。

类似临界区对象，事件对象使用前也需要向系统请求资源句柄，使用后必须释放。不同的是，系统使用 SetEvent 或者 PulseEvent 设置一个事件，使用 ResetEvent 还原这个事件的状态。

③ 互斥器（Mutex）

互斥器的拥有者只能有一个进/线程，互斥器是个同步对象。当没有任何进程占用这个互斥器对象时，该互斥器处于 non-signaled 状态，否则立即进入 signaled 状态。

当使用 CreateMutex 申请资源成功时，进程就拥有了该资源的控制权，别的同样申请该资源的进程就进入等待队列，知道这个进程调用 ReleaseMutex 释放该资源。

④ 信号量（Semaphore）

信号量基本原理和互斥器一样，但是它允许特定数目的进程共享该资源，获得和释放的函数分别是 Create Semaphore 和 Release Semaphore。

信号量是通过维护引用计数起来实现进程间同步的同步对象。他的计数器范围可以从 0 到一个设定的最大值。当对象的计数超过 0 的时候，对象处于 signaled 状态，否则处于 non-signaled 状态。信号量的这种特性通常用于线程的同步或者互斥操作，通过信号量的 PV 操作可以实现进程或者现成的同步。

3.3 Windows CE 的文件系统概述

Windows CE 提供了三种类型的文件系统：RAM-based 文件系统，

ROM-based 文件系统以及支持 ATA 设备和 SRAM 卡等外围设备的 FAT 文件系统。前两种属于 Windows CE 的内建文件系统，后者属于可安装行文件系统。

Windows CE 的文件系统是非常灵活的模块，所有的文件系统以及文件操作相关的 API 都由模块 filesys.exe 提供。该模块实现了对象存储和存储管理的功能。所有的文件和文件管理系统都存在于一个以“\”开始的命名空间下，所有文件都位于从根文件目录开始的一棵树，以路径作为唯一标识符。

如图 3.1 所示，filesys.exe 由以下几个组件构成：

- ROM 文件系统
- 存储管理器
- 对象存储

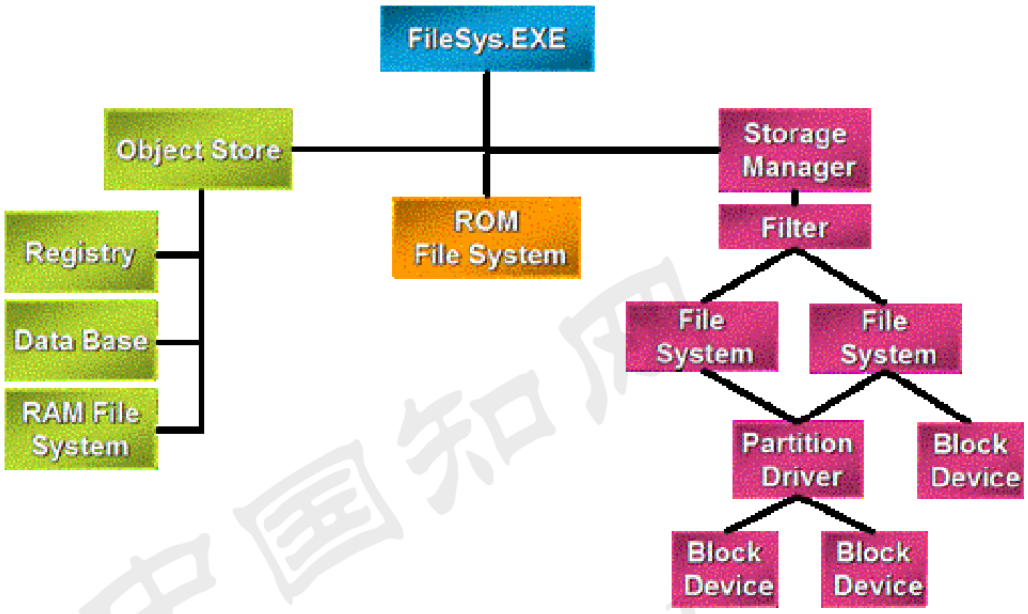

图 2.2 Windows CE 的文件系统的驱动体系结构

对象存储是一种由 filesys.exe 控制的内存堆，它包括了 RAM 中的系统注册表，RAM 文件系统和属性数据库；ROM 文件系统表示了 Windows CE 中以只读形式访问的文件；存储管理器是 WindowsCE.NET 新增的模块，负责管理系统中的存储设备以及使用这些设备所必须的文件系统。

3.4 Windows CE 的内存管理概述

Windows CE 提供灵活的存储器存取机制，使得系统中不同类型的应用程序可以充分的使用系统提供的 RAM，ROM，以及 Flash Memory，并且选择性的有效利用存储器提供的虚拟存储器，存储器保护等功能。

存储器管理可以分为三类：

实体页面管理：主要负责追踪实体存储器的使用情况，为换页程序选取可用的实体页面，释放不使用的实体页面等等。

虚拟存储器管理：主要管理系统的存储器位址对应页面的换进换出等。

堆管理：主要管理处理程序空间内部的动态存储器的释放和回收，以支持程序的动态数据结构。

系统的 32 位虚拟位地址提供了 4GB 的虚拟存储器空间，系统同时处理的程序数目最多 32 个，每个处理程序实际可用的存储器空间受到限制（32MB）。

Windows CE 的内存分配情况如下图 3.2 所示。

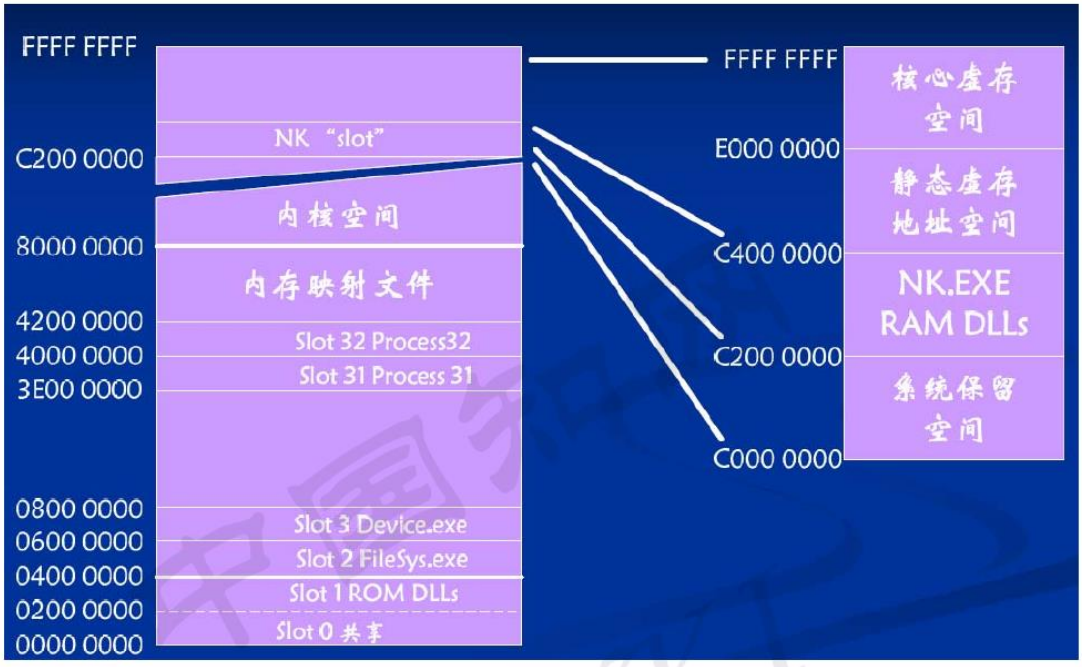

图 3.3 Windows CE 的内存空间分布

系统提供了 33 个 slot 供系统处理的程序使用，其中 slot0 为全域共享内存，底部的一些 slot 由重要的系统处理程序使用，kernal 部分的静态虚拟地

址专门用来对应 ROM，外围设备等资源；NK slot 用来存放 NK.EXE，等等。

Windows CE 只能管理 512MB 的物理内存和寻址 4GB 的虚拟内存。其中这 4GB 的虚拟内存主要划分为两部分，从 0x80000000 的 2GB 以上为内核区，0x80000000 以下的 2GB 为用户态进程使用空间。

关于 Windows CE 具体的的虚拟空间划分，如下表 3.2 所示。

表 3.2 Windows CE **的虚拟内存划分**

地址范围	用途
0x0000 0000~0x41FF FFFF	有所有的应用程序使用，共 33 个 slot，每个 slot 占用 32MB。slot 0 由当前进程使用，slot 1 由 XIP DLL 使用，其他 slot 供其他进程使用，每个进程占用一个 slot
0x4200 0000~0x7FFF FFFF	所有应用程序共享，当 32M 进程地址空间不能够满足的时候，进程就可以使用这个玩飞的地址空间。
0xA000 0000~0xBFFF FFFF	内核重复定义 0x8000 0000～0x9FFF FFFF 之间定义的物理地址映射空间。
0xC000 0000~0xC1FF FFFF	内核保留空间
0xC200 0000~0xC3FF FFFF	NK.EXE 使用的地址空间
0xC400 0000~0xDFFF FFFF	用户使用的虚拟地址空间
0xE000 0000~0xFFFF FFFF	内核使用的虚拟地址空间

[参考文献]

[1]段刚刚.嵌入式操作系统的架构分析与导航应用的实现[J].哈尔滨工业大学,2008 年.

嵌入式系统大作业

学生姓名：文华

学　号：2017218007

专　业：物联网工程

班　级：2017 级 2 班

选题 ：嵌入式系统中的 ARM 指令集与 PC 计算机中的 X86 指令集的区别与联系

指导教师：丁贤庆

完成日期：2019 年 12 月 6 日

目 录

正文

0.

所谓指令集，就是CPU中用来计算和控制计算机系统的一套指令的集合，而每一种新型的CPU在设计时就规定了一系列与其他硬件电路相配合的指令系统。CPU的指令集从主流的体系结构上分为精简指令集（RISC）和复杂指令集（CISC）。嵌入式系统中的主流处理器——ARM处理器，所使用的就是精简指令集。而桌面领域的处理器大部分使用的是复杂指令集，比如Intel的X86系列处理器。我们把ARM处理器所使用的指令集称为ARM指令集，把X86处理器所使用的指令集称为X86指令集。由于ARM处理器与X86处理器采用不同类型的指令集，因而造成了处理器在性能、成本、功耗等方面的诸多差异。现从ARM指令集和X86指令集的特点、操作、功能方面做一比较，以说明两种处理器有诸多差异的原因。

1.

第一，X86指令集随着计算机的功能越来越强大，计算机内部的元件越来越多，指令也相应的变得十分复杂，而在使用过程中，并不是每一条指令都要完全被执行，在技术人员的研究过程中发现，约有80%的程序只用到了20%的指令，而一些过于冗余的指令严重影响到了计算机的工作效率。而ARM指令集种类大大的减少，指令只提供简单的操作，使一个周期就可以执行一条指令。编译器或者程序员通过几条简单指令的组合来实现一个复杂的操作（例如，除法操作）。

2.

第二，由于 X86 指令集是属于 CISC 类型的指令集，其每条指令的长度是不固定的，而且有几种不同的格式，这样一来，就造成了 X86 处理器的解码工作非常复杂。为了提高处理器的工作频率，就不得不延长处理器中的流水线。而过长的流水线在分支如果出现预测出错的情况，又会带来 CPU 工作停滞时间较长的弊端。而 ARM 指令集大多数指令采用相同的字节长度，并且在字边界上对齐，字段位置固定，特别是操作码的位置。这就非常适合采用流水线技术，允许流水线在当前指令译码阶段去取其下一条指令。

3.

第三，X86 指令采用了可访问内存地址的方法，这样的方法容易造成处理器与内存之间的不平衡工作，从而降低处理器的工作效率。而 ARM 处理器则是使用 Load/Store 的存储模式，其中只有 Load 和 Store 指令才能从内存中读取数据到寄存器，所有其他指令只对寄存器中的操作数进行计算。因此，每条指令中访问的内存地址不会超过 1 个，指令访问内存的操作不会与算术操作混在一起。

4.

第四，X86 构架处理器中的 FPU（Floating Point Unit）浮点运算单元的运算能力较差，其主要原因就是 X86 指令集中所使用的一个操作数堆栈。如果在运算过程中，没有足够的寄存器进行计算，系统就不得不使用堆栈来存放数据，这样一来会浪费大量的时间来处理 FXCH 指令，才能将正确的数据放到堆栈的顶部。ARM 处理器本身不支持浮点运算,所有的浮点运算都在一个特殊的浮点模拟器中运行，并且速度很慢，经常需要进行数千个时钟周期才能完成浮点函数的计算。

5.

第五，在流水线方面，ARM 指令的处理过程被拆分成几个更小的、能够被流水线并行执行的单元。在理想情况下，流水线每周期前进一步，可获得最高的吞吐率；而 X86 指令集的执行需要调用微代码的一个微程序，在执行速度上不如 ARM 指令集。

6.

第六，X86 指令对于各种扩展部件的限制也是十分不利的。首先，X86 架构的处理器对于 4GB 的内存容量上限制，虽然现在目前主流的个人电脑的内存大小为 512MB 和 1GB，但是相信随着操作系统和应用软件的不断提升，会快将会突破 4GB 的内存容量。而 ARM 则相反，它可以支持丰富的扩展部件。

7.

第七，为了提高 X86 架构的处理器的性能，而出现像寄存器重命名、缓冲器巨大、乱序执行、分支预测、X86 指令转化等等现象，都使得处理器的核心面积变得越来越大，这也限制了处理器工作频率的进一步提升，设计成本增加，此外，处理器所集成的这些庞大数目的晶体管都只是为了解决 X86 指令的问题。而 ARM 指令集可以大大简化处理器的控制器和其他功能单元的设计，不必使用大量专用寄存器，特别是允许以硬件线路来实现指令操作，从而节约的处理器的制造成本，核心面积小。

8.

第八，ARM 指令集还加强了并行处理能力，非常适合于采用处理器的流水线、超流水线和超标量技术，从而实现指令级并行操作，提高处理器的性能。而且随着 VLSI（Very Large Scale Integration，超大规模集成电路）技术的发展，整个处理器的核心甚至多个处理器核心都可以集成在一个芯片上。然而 X86 指令集却给 VLSI 设计带来很大的设计负担，不利于单片集成。

［参考文献］

[1]金瑶,陈磊萍. ARM 指令集与 X86 指令集之比较[J]. 成功(教育),2007.