TEC-XP＋
计算机组成原理与
系统结构实验系统
教师实验指导书

清华大学科教仪器厂
清华大学计算机系
2007 年 4 月

目录

第一章 TEC-XP＋计算机组成原理与系统结构实验系统概述

1. TEC-XP＋教学计算机系统概述

1.1 TEC-XP＋教学计算机系统系列和总体组成概述

TEC-XP＋是由清华大学计算机系和清华大学科教仪器厂联合研制并通过了教育部主持的成果鉴定的适用于计算机组成原理与系统结构的实验系统，主要用于计算机组成原理和计算机系统结构等课程的硬件教学实验，同时还支持监控程序、汇编语言程序设计、BASIC 高级语言程序设计等软件方面的教学实验。它的功能设计和实现技术，都紧紧地围绕着对课程教学内容的覆盖程度和所能完成的教学实验项目的质量与水平来进行安排。其突出特点有二，一是硬、软件基本配置比较完整，能覆盖相关课程主要教学内容，支持的教学实验项目多且水平高，文字与图纸资料相对齐全。二是既有用不同集成度的半导体器件实现的真实“硬件”计算机系统，同时还有在 PC 计算机上用软件实现的功能完全相同的教学计算机的“软件”模拟系统，其组成和实现的功能如下图所示。

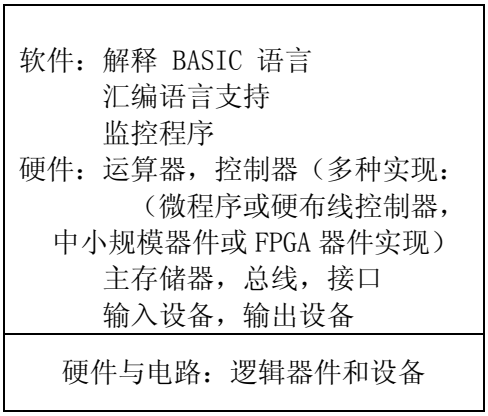


图 1.1 硬件实现的实际计算机系统

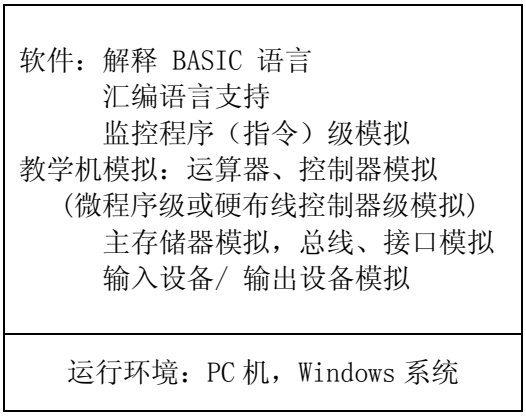


图 1.2 软件实现的模拟计算机系统

从图 1.1 可以看到，该计算机硬件系统组成中，功能部件是完整齐备的，运算器、控制器、存储器、计算机总线、输入输出接口等配备齐全，还可以接通 PC 机仿真终端执行输入输出操作，同时实现了微程序方案的和硬布线方案的 2 种控制器。从 CPU 的具体设计和实现技术区分，既支持用中小集成度芯片实现 CPU 的方案，也支持选用高集成度的 FPGA 门阵列器件实现 CPU 的方案，体现了 CPU 系统设计的最新水平。

从计算机组成原理课程教学实验的角度看，该计算机软件系统组成也是完整的，支持简单的高级语言 BASIC（包括浮点运算指令和基本函数运算功能），汇编语言（支持基本伪指令功能）和二进制的机器语言，配有自己的监控程序，以及 PC 机仿真终端程序等。毫无疑问，全部软件的源程序代码是宝贵的教学参考资料。

从图 1.2 可以看到，软件实现的计算机指令级模拟系统，可以使实验人员脱离实际的教学计算机系统，在 PC 机上执行教学计算机软件系统的全部功能；微程序和硬连线这一级别的模拟软件，可以通过 PC 机屏幕查看在教学计算机内部数据、指令的流动过程，并显示每一步的运行结果，为设计、调试教学机新的软件或硬件功能提供重要的辅助作用。

值得一提的还研制了控制器（微程序或硬连线方案）辅助设计软件，同学可以在 PC 机上使用该软件直接设计该计算机的控制器，包括定义指令格式和编码，划分指令执行步骤和每一步的操作功能，确定控制器需要提供的全部控制信号等全部过程，最后会自动生成能装入教学计算机硬件系统中实际应用的最终结果文件。接下来还可以选用微程序级的模拟软件系统，或者硬布线控制器级的模拟软件系统，对经过辅助设计软件得到的设计结果进行模拟运行，计算机屏幕上会详细显示每一步的运行结果，做到尽早地发现问题。由于在执行上述的控制器设计和模拟运行的整个过程中，都是在 PC 机上完成的，脱离实际的教学机系统，工作更方便，效率更高，对节省学时、帮助同学加深对控制器组成、设计等方面的理解深度也有益处。

1.2 TEC-XP＋教学计算机的指令系统设计

合理地确定一台计算机的指令系统，无论对计算机厂家还是对最终用户来说都是十分重要的事情，它密切关系到计算机设计与实现的复杂程度和生产成本，计算机使用的难易程度和运行效率。对主要用于教学和教学实验目的的计算机，特别是对于一台 16 位字长的教学计算机来说，确定其指令系统，更多地应关注它在教学过程中的作用和使用方法，至少应解决好以下几个问题。

（1）指令格式和功能的典型性，即选择 DLX 指令集结构，适当靠拢 RISC 机的指令格式，做到尽可能小的指令集，简化的寻址方式。这样做不仅使教学计算机的结构简化，实现简单，易于实现指令流水。做到指令格式和功能有良好的典型性，同学更容易接受，讲课时更容易完整地讲解清楚这套指令系统和控制器设计，也有利于教学内容的整体安排。

（2）指令系统要有一定的完备程度，给出的指令格式适当规范，指令分类合理，指令执行步骤容易理解，符合人们通常的编程使用习惯。总之，有较好的易学易用性。确保选用这套指令系统，能方便地设计教学计算机的配套软件。

（3）更高的可扩充性，即为学生添加各种新的指令留下比较充足的余地，为此可以把完整的系统中指令划分为必备的（约 30 条）基本指令（设计者已经实现）和待扩展的（约 20 条）保留指令（留给学生设计实现）2 大类；在扩展新的指令时，实现手段要适当简单，但要有比较多的设计内容和选择余地，以便更好地培养学生的创新意识和开创能力，有利于深化教学内容。

（4）符合教学计算机的特定要求。对 16 位字长的计算机，指令的操作码部分可以选择为固定长度；再结合我们所选用的运算器器件 Am2901 芯片内含 16 个通用寄存器的特点，寄存器寻址方式需要使用 4 位的形式地址。如果需要，还可以指定 16 个累加器中的几个为专用的寄存器，以便最大程度地简化教学机硬件组成，简化指令执行流程设计。

遵照上述思路，最终确定了教学计算机的指令系统的具体组成和指令格式。指令格式如图 1.3 所示。从图中可以看到，指令中包括单字指令和双字指令，第一个指令字的高 8 位是指令操作码字段，低 8 位和双字指令的第二个指令字是操作数地址字段，分别有 3 种用法。

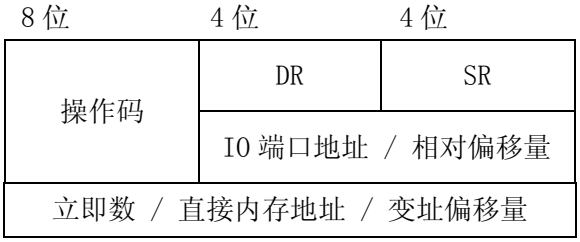


图 1.3　教学机的指令格式

8 位指令操作码（记作“IR15~IR8”），各位的含义如下：

IR15、IR14　用于区分指令组：　0×表示 A 组，10 表示 B 组，11 表示 C、D 组；

IR13　用于区分基本指令和扩展指令：0 表示基本指令，1 表示扩展指令；

IR12　用于简化控制器的实现，暂定该位的值为 0；

IR11~IR8　用于区分同一指令组中的不同指令（最多 16 条）；

IR11　还用于区分 C、D 组指令（每组最多 8 条）：0 表示 C 组，1 表示 D 组。

第一个指令字中的操作数地址字段可以给出：4 位的通用寄存器编号（DR 代表目的寄存器，SR 代表源寄存器），8 位的 IO 端口地址，8 位的相对变址偏移量。第二个指令字用于给出 16 位的立即数，16 位的直接内存地址，或者 16 位的变址偏移量。

1.3 TEC-XP＋教学计算机的结构和组成设计

作为教学和教学实验使用的计算机，其结构和组成设计要比较好地体现出尽可能多的主要教学内容，包括功能部件划分清晰，设计合理，它们之间连接关系适当规范等。

在选用中小规模集成度器件实现的 CPU 系统中，运算器部件设计选用了位片结构的 4 位长度的运算器芯片，内含功能比较合理的 ALU，双端口控制读出、单端口控制写入的 16 个累加器，和完成乘除法运算的乘商寄存器等功能部件，从功能和组成两个方面都比较好地体现了运算器部件的教学内容。

在控制器部件设计中，同时实现了微程序的和硬布线的两种控制器，通过一个开关简单地完成两种控制器之间的切换。讲课过程中，以一种控制器方案为主，对控制器的组成与设计技术讲明讲透，再用少量学时顺便介绍另外一种控制器方案，有利于比较两种控制器的异同之处和各自的优缺点，可以取得事半功倍的教学效果。

设计指令执行步骤时，对选用中小规模集成电路实现的控制器，为了突出基本原理和减少器件数量，指令被设计为串行执行，即下一条指令必须在当前指令完全结束后才能开始；在选用高集成度的现场可编程器件 FPGA 实现的 CPU 系统中，继承了原来的指令系统，既支持指令的串行执行，也可以选用指令流水线技术实现指令的并行执行，以支持真实的指令流水线的教学实验功能。

在教学计算机存储器部件设计中，出于简化和容易实现的目的，选用静态存储器芯片实现内存储器系统，包括了唯读存储区（ROM，存放监控程序等）和随读写存储区（RAM）两部分，也可以实现指令和数据分开的两个存储体（只用于 FPGA 实现的带指令流水线的 CPU 系统）。适当改进后，可以支持存储器的多体交叉编址技术。

在教学计算机总线部件设计中，实现了单总线结构，数据总线、地址总线和控制总线比较简单，保证教学机的正常运行并体现出总线设计的基本原理。

关于计算机中的接口线路，教学计算机提供了 2 路串行接口（INTEL 8251），可以接入 PC 机作为教学计算机的仿真终端完成输入输出操作。

关于中断处理，支持 3 级的中断并允许中断嵌套，可以完成常规的中断处理能力，对中断优先级编码与排队，中断响应和现场切换等处理上特色很强。

1.4　教学计算机的硬件实现技术

作为教学目的的计算机系统，在具体的硬件实现方面与一般的商用计算机有着某些明显不同之处，具体表现在如下 3 个方面。

首先是“处处可见”的要求，不仅全部部件和全部器件要做成全暴露方式，还包括计算机运行过程中各种数据和控制信号的状态及其时空关系也容易观察，这对于学生完成教学实验，掌握计算机组成与设计技术是必不可少的。因此，教学机的线路板上设置了大量的开关和指示灯，适当的跳接线插针、插孔等。随之带来的一个突出矛盾是设备的可靠性和更高的抗破坏性，换句话说，要能够尽量降低学生实验过程时常会发生的错误对教学机的损坏程度，还要容易维修，这是教学计算机设计与实现过程中的难点之一。

其次，在教学计算机实现中，要给出尽可能高的可扩展性，即可以方便地修改已有设计，或者增加新的功能。限制太多，留的选择余地太少，会严重影响教学实验的水平和质量；保留的选择余地过多，不仅难以实现，还会增大教学实验难度，反过来也难以保证教学实验的质量。如何在这二者之间进行取舍，无疑是留给设计者的又一个难题。作为主导思想，我们还是更强调选择余地留得适当多一点，通过加强实验指导，实验说明写的详细一些，使得学生可以开展一些有创意的教学实验内容，全力增强学生的创新意识和自主学习的能力。

最后一点是通过教学计算机系统能够体现出多大的知识面，即涵盖多大范围的主要教学内容。为了兼顾不同院校的教学安排，针对计算机组成原理和计算机体系结构两门课程的实验需求，我们把教学计算机系统做成了一个系列产品。

从计算机字长划分，可以实现 8 位字长的计算机，也可以实现 16 位字长的计算机，难能可贵的是它们是在同一个印制电路板上实现的，换言之，只要教师具备相应的知识和技术能力，在 16 位的教学计算机上，经过重新设计并组装成 8 位字长的计算机是可行的，当然这已经是完全不同指令系统、不同硬软件组成的两套计算机，可以用在计算机组成原理课程的教学过程中，用于课程设计，甚至于作为学生的毕业设计题目也是可行的。

从选用的器件划分，支持用中小规模集成电路实现的 CPU 系统，在体现计算机组成的原理性知识的同时，在很多场合都要涉及到数字逻辑电路和逻辑设计的具体知识，更接近计算机实现的线路逻辑的底层，体现了比较传统的教学内容和实验安排。还支持用大规模集成电路 FPGA 芯片实现的单个芯片的 CPU 系统，选用 VHDL 语言描述计算机处理器的功能组成，此时则较少涉及计算机具体实现的线路逻辑，更有利于突出计算机的功能设计与实现，并且为实现指令流水提供了最好的条件，体现出比较新的教学内容和更高层次的实验安排。

从计算机系统的完整性划分，目前国内多数的计算机组成原理课程的教学实验是在硬件裸机装置上完成的，实验装置上没有配备合理的软件系统，计算机组成原理被作为一门“纯正的”硬件课程来看待，这是很过时的做法，与国外先进的教学安排相差甚远。我们则坚持把一台硬软件配置比较完整的计算机系统用于教学实验，汇编语言程序设计、简单的操作系统雏形（监控程序），不同层次的软件模拟等应该在计算机组成原理课程中有适当的地位。

从计算机硬件系统的设计手段划分，在教学计算机设计与实现中，有非常老式的、基本上完全手工设计的例子，也有更现代化的通过 VHDL 语言描述、并用 FPGA 芯片实现的例子。使同学们有机会在二者之间进行某些对比，看到技术进步给 IT 产业带来的深远影响。

除此之外，我们还设计了计算机控制器（含微程序控制器和硬连线控制器）辅助设计软件，该软件将以导航方式指明控制器的设计流程，引导学生逐项完成必要的设计，包括定义指令系统，划分指令执行步骤和每一步实现的功能，选择有关控制信号等，并把原来完全手工设计过程中那些机械重复的微指令编码、微指令地址安排，以及书写硬连线控制器每一个控制信号的逻辑表达式等烦人的劳动交由计算机自动完成，极大地提高了教学实验的工作效率和设计质量。接下来可以通过运行模拟软件系统，检查辅助设计的结果，尽快地帮助找出设计中的错误。正确的设计结果可以被直接装入到实际的硬件系统中，通过运行最终设计的硬件教学计算机系统，查看实际运行效果。设计

过程和操作运行步骤密切相关，环环紧扣，使学生在探索中学习，浓厚的兴趣，油然而生的成就感将激励他们不断前进。

1.5 软件模拟实现的教学计算机系统

把已经用硬件实现的教学计算机系统的全部功能，通过软件模拟的办法在 PC 机上再次展现出来，是我们做出的重大努力之一。通过网上查询和与已经出国人员的直接交流，了解到国外有一些著名大学在计算机组成原理课程教学过程中的安排，多选用软件模拟的方式完成逐项教学实验。其优点是使用方便，变动设计的灵活性强，可以比较容易地对比不同设计方案对计算机执行性能影响的程度。不使用专用硬件设备完成教学实验，实验成本也会比较低，不要求专职的人员管理硬件教学实验室，授课与辅导教师更容易与同学交互。缺憾也是明显的，学生学习硬件课程的整个过程，不接触（拥有）自己可以设计与修改的硬件设备，更多的精力集中到计算机的功能设计部分，难以对线路与逻辑设计部分，计算机硬件实现中的工程性、技术性问题有切身体会。如何权衡硬件的教学计算机系统和软件模拟的教学计算机系统在课程教学中的作用，可能是仁者见仁，智者见智，恐怕一时也难有定论。对此，我们采取的措施是同时实现出这样两部分内容，并且同时使用在教学过程中，使它们发挥各自不同的作用，通过教学实践来探索更好的解决问题的途径。为了更便于比较，做到更好的资源复用，在设计与实现软件模拟的教学计算机系统的过程中，采取了两项措施：

（1） 坚持与硬件实现的教学计算机系统有尽可能高的一致性。为此，模拟软件使用的信息，例如监控程序的执行码，微程序控制器的微程序的二进制编码文件等，与硬件教学机系统中使用的完全相同，这样辅助设计产生的设计结果，既可以直接用于模拟，也可以直接用于写到硬件教学机的部件中，确保二者之间有最好的一致性。

（2） 在模拟软件的设计中，比较准确地按照硬件系统的主要功能部件实现模拟，确保硬件实现的与软件模拟现实的系统有良好的对照关系，有望在教学过程中可以得到更好的教学效果。

实现的模拟系统，模拟重点分配到 2 个层次上。

一个层次是指令级模拟，模拟的最小单位是一条机器指令,根据得到的指令的具体内容计算出这条指令的执行结果。这个模拟软件的运行对象是机器指令组成的程序，运行结果将显示在计算机的屏幕上。当被运行的程序对象是监控程序时，实际上就是在模拟监控程序的执行过程，可以执行每一个监控命令，与在真实教学计算机上运行监控程序的效果是一样的，此时也可以说是在执行教学计算机系统一级的模拟，可以建立并运行用户的汇编程序，也可以建立并运行用户的 BASIC 语言的程序等。

另一个层次是微体系结构级模拟，模拟的最小单位是一条指令的一个执行步骤，对微程序控制器而言是一条微指令，对硬连线控制器而言是指令的一个节拍。这个模拟软件的运行对象也是机器指令组成的程序，运行结果将显示在计算机的屏幕上，但它有 2 种不同的运行方式：连续运行和单步运行。在连续运行方式下，在计算机屏幕上见到的运行效果和前一个层次上的运行效果是一样的，只是运行速度要慢得多，因为要模拟计算每一条指令的每一个执行步骤的结果，经过几个模拟步骤才能得到一条指令的运行结果，承担模拟的 PC 计算机的计算量要增大很多。在单步运行方式下，在计算机屏幕上将显示指令的这个执行步骤的有关运行结果，包括指令寄存器的内容，指令所处的执行步骤，地址总线、数据总线、各个累加器的内容，运算器的输出结果和标志位的结果，状态寄存器的内容，控制器全部控制信号的当前状态等完整的信息，主要用于查看、理解、分析计算机内指令执行的详实过程，这对检查同学新增加的控制器功能（扩展的新指令的执行过程）的正确性是很有帮助的。

开发模拟软件需要注意选择运行平台。考虑到已经把 Windows 系统作为仿真终端的运行环境，全部软件系统也都统一到这一平台之上。

1.6 教学计算机在教学过程中的作用

TEC-XP+教学计算机系统的组成和设计技术，作为教学内容的实例有很强的典型性，修改或扩展已有功能可以比较好地满足教学实验的要求；硬件实现的计算机系统可以用于现场的真实硬件系统的教学实验，软件模拟实现的计算机系统，可以在脱离实际教学计算机系统的条件下，在 PC 机上通过模拟的办法完成原本必须在教学机上进行的教学实验。

计算机组成原理是计算机系本科生最重要的专业基础课程之一，必修，课程在整个教学安排中占有重要的地位。计算机系统结构课程也是计算机系本科生重要的专业课程之一，但系统结构课程的教学实验却不容易落实，目前比较多地集中到通过软件模拟来评测不同设计方案对系统性能的影响。应该清醒地看到，国内对于计算机硬件为主的课程的定位尚不够准确，教学过程中也还存在一些模糊认识，或由于条件上的限制采取了一些不够进取的办法，这一切严重地影响教学质量，长远看，对我国信息产业的发展是不利的。

对硬件为主的课程定位不够准确，主要表现在一种“欺软怕硬”的心理，认为计算机硬件（尤其是计算机组成原理课程）难教更难学，于是采取适当躲避的态度对待之，极端一点的，直接说计算机组成原理课程没有用。显而易见，如果计算机专业的本科生，连计算机组成原理这样一些专业基础知识都不具备，如何更好地学习后续的专业课程，又如何能发展成为计算机的专业人士呢？更不要说是高级专门人才。

对课程的模糊认识之一，认为计算机组成原理是纯硬件课，把它和计算机软件系统决然分开，而不是把它作为完整计算机系统的组成部分之一来合理安排教学内容，这是国外 30 多年前就已舍弃的陈旧做法。模糊认识之二，认为计算机基础硬件不再重要，以为主要应该讲解高档计算机中最新出现的技术和最新进展，殊不知这些最新出现的技术和最新进展都是在基本原理上发展而来的，轻视专业基础知识，最终架设的知识结构只能是空中楼阁，更不符合大学通才教育的要求。

条件上的限制，主要表现在缺乏必要的、能比较好地满足教学实验要求的教学实验设备，使教学过程更多地建立在课堂授课的方式下，缺乏必要的教学实践支持，这对于技术性和工程性很强的课程来说几乎成为灾难性的问题。引申一步看，还包括部分任课教师所具有的知识结构基本上也只限于书本知识，缺乏设计和实现一台真实一点的计算机基本硬、软件系统的经历，一时还难以完全适应教学的实际需要。

我们的观点是，解决上述问题的出路之一，在于建设被广泛接受和认可的、高质量的教学资源，包括体现最新水准的高质量的教材，形象生动的教学课件（教学教案），研制并合理配备能覆盖主要教学内容、具有良好教学实验功能的教学实验设备和软件模拟系统。

在清华大学计算机系，“计算机组成原理”课程教学改革的思路是，从课堂教师讲授为主、学生被动地听课和完成验证性的教学实验为辅的传统“知识传授型”的教法，改变成为在精简课堂授课内容的基础上，以完成一项计算机完整的硬、软件系统的工程设计与实现的课题研究为重要支撑手段的“学生主动探索式”新的教学途径，尽量做到：

从学生听课为主的被动接收知识的学习方式，变成为结合课题研究与工程设计来主动学习的重大变革，给学生更多自由支配的时间，充分发挥学生自主学习的积极性；

从传授知识为主变为传授知识与培养能力并重的人才培养模式。清华大学关注开创意识和创新能力的培养，注重学生全面素质的提高。特别是考虑到计算机组成原理课程教学内容的工程性、技术性、实践性特别强的特点，把工程设计项目引进教学全过程很有必要。

教学计算机系统在清华大学计算机系的教学改革中显示出强大的生命力，在为期一个学期的计算机组成原理课程的教学过程中，要求每 3 名同学为一组，合作完成一台与教学实例有明显差异的、硬软件配置基本完整的全新计算机系统，包括运算器、存储器、微程序的控制器、硬布线的控制器等部件设计与实现，也包括适当修改监控程序和汇编器软件等软件设计任务，这是一项很有挑战性的任务。针对清华大学创办“世界一流”和“研究型”大学的目标，这种努力是值得的。目前清华大学计算机系的“计算机组成原理”课程已经被评为北京市精品课和国家级精品课，并荣获北京市优秀教学成果一等奖，教材被评为北京市精品教材。在全国，已有几百余所大学把我们略早期研制的中小规模器件实现的硬件教学计算机系统用于他们的授课过程和教学实验，反响还相当不错。我们也知道，在清华大学计算机系的开展的教学改革方案，不能简单地在全国各院校直接照搬，对大部分的教师来说，他们可能更倾向于我们前一些年一直采用的比较传统的做法，就是通过在教学计算机上做一些验证性为主的实验，或者适当扩展教学计算机已有功能的教学实验。即使这样，也在原来的基础上前进了一大步。在这一改革方案中，首先受益的是年轻教师，只要他们肯花出适当的努力，就可以填补上所欠缺的设计与实现完整计算机的基本硬、软件系统的经历，提高自己的业务素质和教学水平。接下来学生将全面受益，课程内容中既有适量的书本知识，教师也会结合教学机设计增加实例内容，课后学生又可以在教学机上完成自己的设计和调试任务，力争做到学习知识与增长能力的双丰收。

此外，对技术含量更高，对教学影响可能更大一些最新的研制成果，包括控制器辅助设计系统，全部的软件模拟系统，用 FPGA 芯片实现的 CPU 系统（与原有教学计算机系统指令兼容，并增加了比较真实的指令流水线），高级语言支持和浮点运算指令支持等，由于推出时间较晚，了解的人还不多，有必要把这些内容尽早地引入教学过程，推动课程的教学改革的深入发展，培养出更多具有创新能力的高素质的专业人才。

1.7 基本实验项目设置

在本节提到的实验项目，多数是在 TH-union 系统中选用中小规模集成度器件实现的教学计算机上可以开设的实验，把选用 FPGA 芯片实现的 CPU 构建的教学机系统中支持的教学实验项目放在第 3 章介绍。

基本实验是指学习计算机组成原理通常总要完成的实验项目，解决的是学习基本原理和培养基本能力的问题。

1. 基础汇编程序设计学习

使用系统已实现的 29 条指令和监控程序、交叉汇编程序软件，设计与调试由教师布置的或自己设想的各种汇编程序。如有可能，可以参照系统提供的交叉汇编程序的源码，学习系统汇编程序(Assembler)的实现原理与设计技术。

2. 运算器部件实验

可以在运算器完全脱开主机控制的方式下，用主板上的微型开关直接控制运算器的方式来使用运算器并观察运算结果；也可以在实验计算机正常运行方式下，用控制器给出实验者所设计的对运算器的控制信号来使用运算器并观察运算结果（此时并不需要懂得控制器的运行原理）。

可以观察并量测运算器在串行进位方式下的进位信号的波形和延迟时间，也可以测出该运算器所允许的最高工作频率。

3. 主存储器部件的实验

可以进行存储器工作波形的观察与量测，可进行静态存储器的容量扩展实验，通过监控命令或自己设计的小程序向存储器写入数据并检查读写的正确性。

4. 控制器部件实验

可以做微程序方案或硬布线方案的控制器实验。首先是通过听课和操作实验机，学懂已实现的控制器的设计原理与实现技术，以单指令方式、单步骤执行方式观察指令的运行结果和信息在计算机内产生和传送的时间、空间关系，这是更重要的实验内容；然后才是设计与实现多条自己定义的新指令，并把新老指令放在同一程序中运行，检查结果的正确性。在微程序方案下实现新指令更容易，不用改动任何硬件，按规定办法把有关新指令用到的微程序装入控制存储器即可。在硬连线控制器方案中，则要把新指令用到的控制逻辑与原有的基本指令的控制逻辑合并一起，经过编译之后再重新下载到可编程的 MACH 器件中，略显得复杂一点，可能要多次地改正错误才能得到最终的正确结果。

5. 串行口输入/输出实验

由于本机上已给出了两路串行接口，其中的一路的接线完全连接好，系统也已经执行了对接口芯片的初始化，可以直接用于输入输出操作；另外的一路的接线并未全部连通，要求实验人员看懂图纸并完成必要的连线操作和串行口的初始化操作后，方能用程序控制方式完成该串行接口的正常输入/输出操作，例如用两台实验机的这一路串行口完成双机双向通信等实验。作为更高要求，还可以增加少量硬件线路，实现在中断方式下完成输入/输出操作(需在讲过中断之后进行)。还可以观察与量测串行数据的波形，起始位、停止位、串行数据采样时间的配合关系等。

6. 并行口与并行口打印机驱动的实验

可以在教学机的主板上设计并搭建诸如 Intel 8255 并行接口与配套逻辑电路，并用程序控制方式和中断方式驱动并行口打印机完成打印操作。在无打印机的情况下，可用并行口在程序控制方式下或中断方式下实现两台实验机的双机单向或双机双向通信，或在同一台实验机上实现内存内容"搬移"等实验。这涉及硬件与软件两个方面知识的综合应用。

7. 中断及 DMA 实验

串行口、并行口输入/输出操作中，都可以有中断方式下的输入输出方式，这需要适当地修改监控程序。这里也可以专门做多级中断、优先级排队及中断嵌套的实验。此时可用按钮等作为中断请求信号来源，抛开相应设备入/出以强化中断处理本身的份量。

DMA 的工作方式，最简单的实验是在程序正常运行的同时，用 DMA 方式同时完成内存内容的"搬移"操作，结果的正确性便于检查。这个实验需用到扩展的通用接口芯片的插座，完成必要的连线，设计实验小程序软件等。

8. 整机故障定位与排除实验

实验机上有一些跳线夹，是用来人为设置机器故障的。教师可以通过移走一或几个跳线，或换上有故障的器件，要求学生发现故障，查清故障原因并设法排除。该实验有一定难度，但也是综合应用所学全部知识、完成能力训练的非常有效的手段，对提高分析问题与解决问题的能力会有很大帮助。此时需要确保设置的故障不会损坏教学计算机系统的硬件。

1.8 其他实验项目

是指那些不一定是学习计算机组成原理课程必做的实验，或难度更大的实验项目。

1. 故障诊断软件的设计与实现

这可以在机器指令级或微体系结构级进行。机器指令级的诊断程序用以疹断指令与监控程序运行的正确性，是机器出厂前例行实验的一部分，也是实验指导教师把实验机交付学生使用之前判断机器好坏的简便手段。可以让学生试着设计。

微体系结构级的诊断，设计得好的话，可用来实现实验机的故障诊断，非常有用，对于透彻掌握实验机的组成与运行机制帮助巨大，但在教学机中实现起来有相当的难度。

2. 用一台正确运行的实验机辅助调试另一台实验机

在本实验机的实现过程中，已考虑到这类实验的需求。可以把同一时钟同时提供给两台实验机，使其完全同步运行，并比较检查两台实验机内部主要信号、运行状态及结果的一致性，并依此结果判断是否继续给出后续时钟，则很容易找到待调试计算机的出错位置。与此类似的是双机同步运行，是检查机器可靠性的一条捷径。

3. 实验机的监控程序、交叉汇编程序的修改与扩充功能的实验。

4. 扩充输入/输出接口、设备与驱动程序的实验。

5. 设计与实现一套全新指令系统的 CPU。指令格式可以突破现在规定，寻址方式也可变化，以 16 位字长的一字或多字指令为宜。微指令格式可变，但字长在 56 位以下最方便。运算器也可适当变动，用 Am2901 实现其他型号的运算器功能，或用可编程器件设计一个新的运算器均可行。例如，在这个硬件主板上设计并实现一个全新的 8 位字长的计算机系统，指令格式、寻址方式、监控程序等全部软件有关的内容完全重新设计，运算器、控制器、存储器、总线和接口等硬件也完全重新设计，这是完全可行的，工作量是大一些，但可学内容更丰富。对大部分院校来说，这作为学习过计算机组成原理课程之后的一个大的课程设计、甚至于是毕业设计的题目可能更合适一些。

6. 软盘驱动器的接口与驱动线路实验

设计与实现软盘驱动器的接口与驱动线路，是综合应用有关软件与硬件知识的途经之一，可以在教学实验计算机现有功能的基础上，增加新的设备和操作功能，在更大程度上拓展其实验性能。

7. 使用现场可编程器件（GAL20V8 和 MACH）完成组合逻辑的或者时序逻辑的线路实验，对于在学习本课程之前尚未学习过数字电路与逻辑设计课程的学生是必要的。

8. 通过使用第 2 路串行接口和修改监控程序，实现 2 个用户同时操作同一台教学计算机的多用户系统的功能。

以上提出的只是教学计算机系统可以支持（能够实现）的、可以开设的实验项目的思路与可行性，至于更具体地安排哪一些教学实验项目，实验的具体目标、内容、要求等将在下一节中分别详细说明，最终还是应该依据各个单位的教学安排和总体目标由任课教师来具体选择落实。

为减轻教师的备课负担，特给出计算机组成原理若干基本实验的操作步骤，以供教师参考，本书给出的实验操作步骤不一定是最佳方案，教师可自行设计更好的实验和操作。由于篇幅的限制，对于那些需要教师根据实际情况安排的实验没有给出参考步骤，教师可将一部分难度较大需较长时间完成的实验用作计算机组成原理课的课程设计题目或作为学生毕业设计的课题。由于编者水平所限，错误和疏漏之处在所难免，热忱欢迎广大教师和同行提出宝贵意见和建议。

第二章 实验指导

2.1 基础汇编语言程序设计

实验目的：

1．学习和了解 TEC-XP＋教学实验系统监控命令的用法；
2．学习和了解 TEC-XP＋教学实验系统的指令系统；
3．学习简单的 TEC-XP＋教学实验系统汇编程序设计；

实验内容：

1．学习联机使用 TEC-XP＋教学实验系统和仿真终端软件 PCEC.
2．使用监控程序的 R 命令显示/修改寄存器内容、D 命令显示存储器内容、E 命令修改存储器内容；
3．使用 A 命令写一小段汇编程序，U 命令反汇编刚输入的程序，用 G 命令连续运行该程序，用 T、P 命令单步运行并观察程序单步执行情况；

实验要求

在使用该教学机之前，应先熟悉教学机的各个组成部分，及其使用方法。

实验步骤

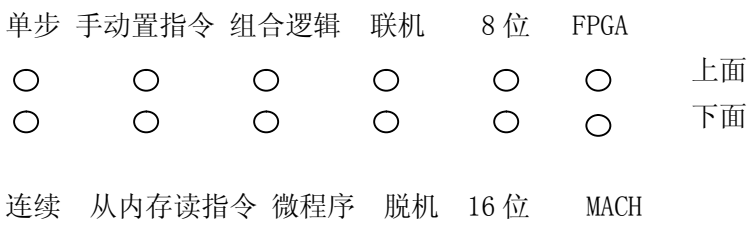
一．实验具体操作步骤：

1．准备一台串口工作良好的 PC 机；
2．将 TEC-XP＋放在实验台上，打开实验箱的盖子，确定电源处于断开状态；
3．将黑色的电源线一端接 220V 交流电源，另一端插在 TEC-XP＋实验箱的电源插座里；
4．取出通讯线，将通讯线的 9 芯插头接在 TEC-XP＋实验箱上的串口“COM1”或“COM2”上，另一端接到 PC 机的串口上；
5．将 TEC-XP＋实验系统左下方的六个黑色的控制机器运行状态的开关置于正确的位置，在这个实验中开关应置为 001100（连续、内存读指令、组合逻辑、联机、16 位、MACH），控制开关的功能在开关上、下方有标识；开关拨向上方表示“1”，拨向下方表示“0”，“X”表示任意，其它实验相同；
6．打开电源，船形开关和 5V 电源指示灯亮。
7．在 PC 机上运行 PCEC16.EXE 文件，根据连接的 PC 机的串口设置所用 PC 机的串口为“1”或“2”，其它的设置一般不用改动,直接回车即可。(具体步骤附后)
8．按一下“RESET”按键,再按一下“START”按键，主机上显示：

```
TEC-2000 CRT MONITOR
Version 1.0 April 2001
Computer Architectur Lab.,  Tsinghua University
Programmed by He Jia
>
```

二、实验注意事项：

1. 连接电源线和通讯线前 TEC-XP＋实验系统的电源开关一定要处于断开状态，否则可能会对 TEC-XP＋实验系统上的芯片和 PC 机的串口造成损害；
2. 六个黑色控制开关的功能示意图如下：

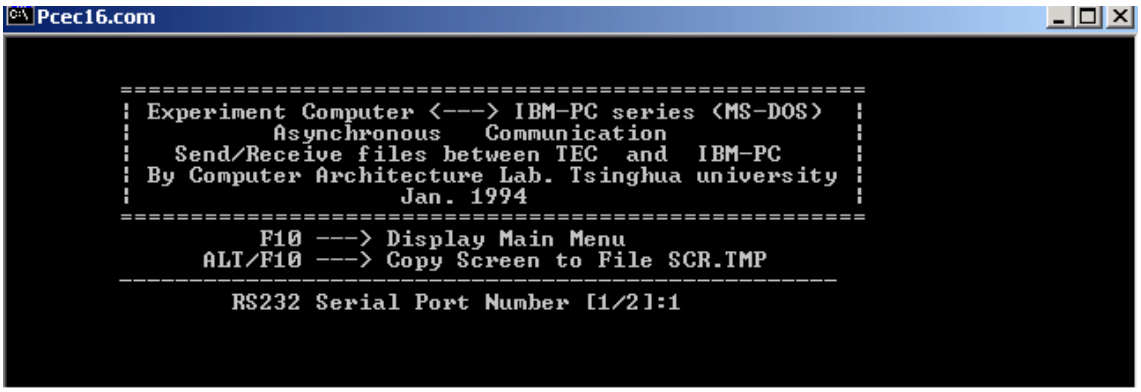


3. 几种常用的工作方式（开关拨到上方表示为 1，拨到下方为 0；）

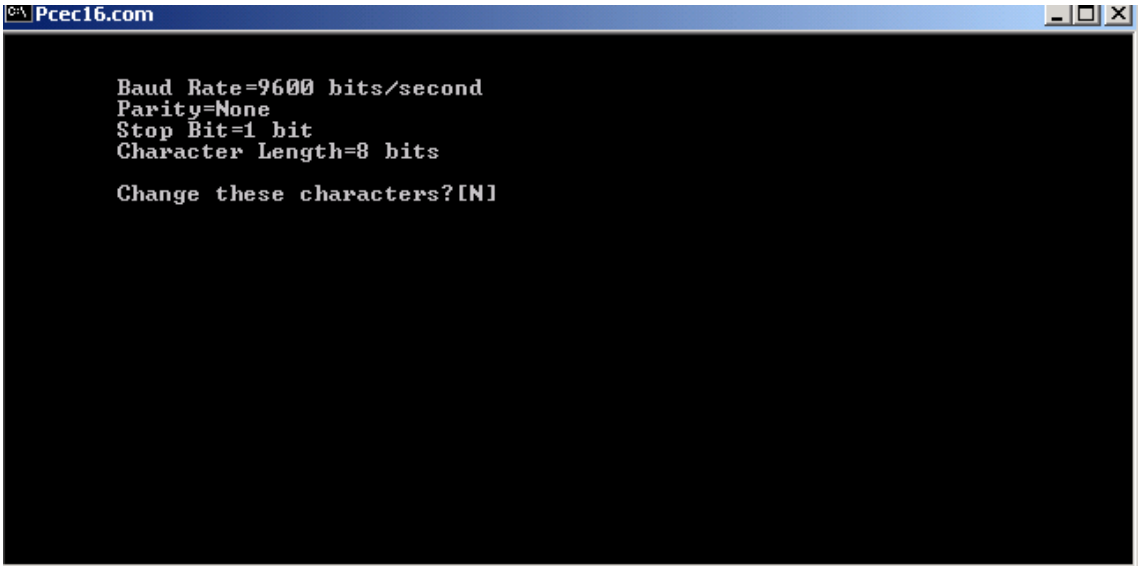
工作方式	六个拨动开关
连续运行程序、组合逻辑控制器、联机、16 位机、MACH	001100
连续运行程序、微程序控制器、联机、16 位机、MACH	000100
单步、手动置指令、组合逻辑控制器、联机、16 位机、MACH	111100
单步、手动置指令、微程序控制器、联机、16 位机、MACH	110100
16 位机、脱机运算器实验、MACH	1XX000
连续运行程序、联机、16 位机、FPGA	00X101

三、仿真终端软件的操作步骤：

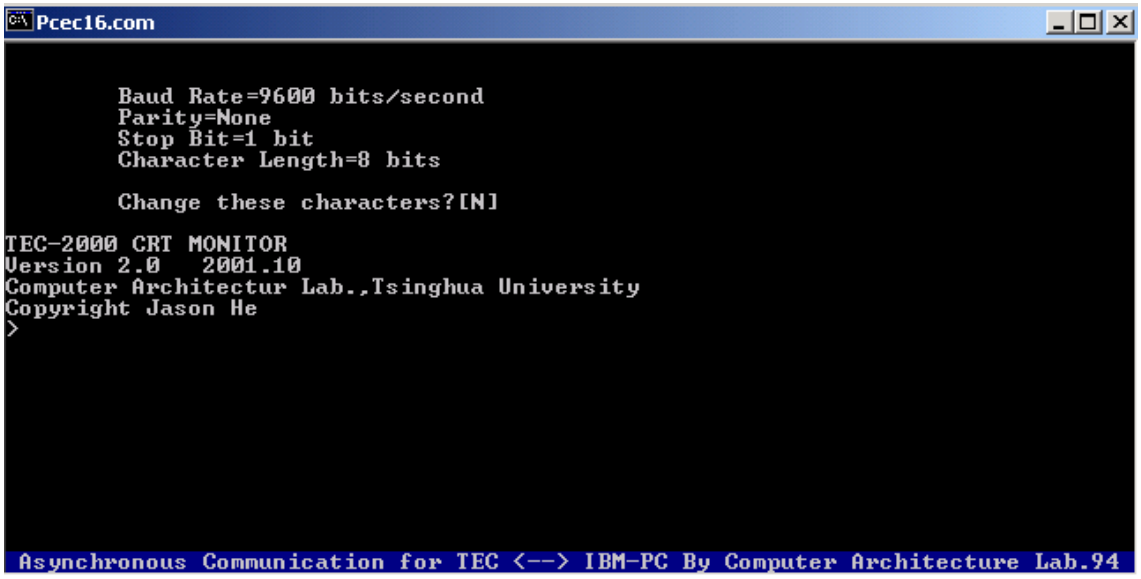
1. 在 PC 机上建一个文件夹 TEC-XP＋；
2. 取出配套的用户盘，将应用程序 PCEC16 拷贝到用户机器硬盘上该文件夹里；
3. 双击 PCEC16 图标，出现如图所示的界面：



4。系统默认选择串口 1，用户可根据实际情况选择串口 1 或是串口 2（这里的串口指的是和 TEC-XP＋教学实验系统相连的 PC 机的串口），按回车后出现如图界面：



5. 图中是系统设定的一些传输参数，建议用户不要改动，直接回车。按一下“RESET”按钮放开后再按一下“START”按钮，出现界面如图所示：



6. 此时表明 TEC-XP＋机器联机通讯正常。

四．软件操作注意事项：

1. 用户在选择串口时，选定的是 PC 机的串口 1 或串口 2，而不是 TEC-XP＋实验系统上的串口。即选定的是用户实验时通讯线接的 PC 机的端口；
2. 如果在运行到第五步时没有出现应该出现的界面，用户需要检查是不是打开了两个软件界面，若是，关掉其中一个再试；
3. 有时若 TEC-XP＋实验系统不通讯，也可以重新启动软件或是重新启动 PC 再试；
4. 在打开该应用软件时，其它的同样会用到该串口的应用软件要先关掉。

五．联机通讯失败自检：

如果上述的硬件和软件的操作都正确，联机却依旧失败，用户可以进行如下测试：

1. 测试 PC 机的串口是否能正常工作，或是换一台 PC 或换同一台 PC 的另一个串口再试，在换串口时要将 TEC-XP＋实验系统断电，换完后重新启动实验系统和软件；
2. 检查机器上的元器件插接是否正确（建议用户对照能够正常通讯的实验系统进行详细检查），有没有被学生动过，尤其是扩展内存和扩展 I/O 接口时，芯片方向是否插对，片选信号有没有连接；
3. 检查相应的短路子是否连接正确；
4. 建议教师预留一台运行正常的 TEC-XP＋实验系统备用，机器出问题后可以对照检查。

六．实验示例：

1. 用 R 命令查看寄存器内容或修改寄存器的内容

1）在命令行提示符状态下输入：

R↙　　；显示寄存器的内容

注：寄存器的内容在运行程序或执行命令后会发生变化。

2）在命令行提示符状态下输入：

R R0↙　；修改寄存器 R0 的内容，被修改的寄存器与所赋值之间可以无空格,也可有一个或数个空格

主机显示:

寄存器原值:_

在该提示符下输入新的值 0036

再用 R 命令显示寄存器内容，则 R0 的内容变为 0036。

2. 用 D 命令显示存储器内容

在命令行提示符状态下输入：

D 2000↙

会显示从 2000H 地址开始的连续 128 个字的内容；

连续使用不带参数的 D 命令，起始地址会自动加 128（即 80H）。

3. 用 E 命令修改存储器内容

在命令行提示符状态下输入：

E 2000↙

屏幕显示:

2000　地址单元的原有内容:光标闪烁等待输入

输入 0000

依次改变地址单元 2001~2005 的内容为:1111 2222 3333 4444 5555

注意：用 E 命令连续修改内存单元的值时，每修改完一个，按一下空格键，系统会自动给出下一个内存单元的值，等待修改；按回车键则退出 E 命令。

4. 用 D 命令显示这几个单元的内容

D 2000↙

可以看到这六个地址单元的内容变为 0000 1111 2222 3333 4444 5555。

5. 用 A 命令键入一段汇编源程序，主要是向累加器送入数据和进行运算，执行程序并观察运行结果。

1）在命令行提示符状态下输入：

A 2000↙；表示该程序从2000H（内存RAM区的起始地址）地址开始

屏幕将显示：

2000：

输入如下形式的程序：

2000: MVRD R0，AAAA ；MVRD与R0之间有且只有一个空格，其他指令相同

2002: MVRD R1，5555

2004: ADD R0，R1

2005: AND R0，R1

2006: RET ；程序的最后一个语句，必须为RET指令

2007:（直接敲回车键，结束A命令输入程序的操作过程）

若输入有误，系统会给出提示并显示出错地址，用户只需在该地址重新输入正确的指令即可。

2）用U命令反汇编刚输入的程序

在命令行提示符状态下输入：

U 2000↙

在相应的地址会得到输入的指令及其操作码

注：连续使用不带参数的U命令时，将从上一次反汇编的最后一条语句之后接着继续反汇编。

3） 用G命令运行前面键入的源程序

G 2000↙

程序运行结束后，可以看到程序的运行结果，屏幕显示各寄存器的值，其中R0和R1的值均为5555H，说明程序运行正确。

4） 用P或T命令，单步执行这段程序，观察指令执行结果

在命令行提示符状态下输入：

T 2000↙

寄存器R0被赋值为AAAAH

T↙

寄存器R1被赋值为5555H

T↙

做加法运算，和放在R0，R0的值变为FFFFH

T↙

做与运算，结果放在R0，R0的值变为5555H

用P命令执行过程同上。

注：T总是执行单条指令，但执行P命令时，则把每一个CALL语句连同被调用的子程序一次执行完成。T、P命令每次执行后均显示所有通用寄存器及状态寄存器的内容，并反汇编出下一条将要执行的指令。

6. 举例编写汇编程序，用“A”命令输入,运行并观察结果

1）例1：设计一个小程序，从键盘上接收一个字符并在屏幕上输出显示该字符。

〈1〉在命令行提示符状态下输入：

A 2000↙ ；

屏幕将显示：

2000：

输入如下形式的程序：

```
2000: IN 81          ；判键盘上是否按了一个键
2001：SHR R0         ；即串行口是否有了输入的字符
2002：SHR R0
2003：JRNC 2000      ；未输入完则循环测试
2004：IN 80          ；接收该字符
2005: OUT 80↙        ；在屏幕上输出显示字符‘6’
2006: RET↙           ；每个用户程序都必须用 RET 指令结束
2007：↙              ；（按回车键即结束输入过程）
```

注：在十六位机中，基本 I/O 接口的地址是确定的，数据口的地址为 80，状态口的地址为 81。

〈2〉用“G”命令运行程序

在命令行提示符状态下输入：

G 2000↙

执行上面输入的程序

光标闪烁等待输入，用户从键盘键入字符后，屏幕会显示该字符。

该例建立了一个从主存 2000H 地址开始的小程序。在这种方式下，所有的数字都约定使用 16 进制数,故数字后不用跟字符 H。每个用户程序的最后一个语句一定为 RET 汇编语句。因为监控程序是选用类似子程序调用方式使实验者的程序投入运行的,用户程序只有用 RET 语句结束，才能保证程序运行结束时能正确返回到监控程序的断点，保证监控程序能继续控制教学机的运行过程。

2）例 2:设计一个小程序，用次数控制在终端屏幕上输出’0’到’9’十个数字符。

〈1〉在命令行提示符状态下输入：

A 2020↙

屏幕将显示：

2020：

从地址 2020H 开始输入下列程序:

```
2020:MVRD R2,000A  ；送入输出字符个数
2022:MVRD R0,0030  ；“0”字符的 ASCII 码送寄存器 R0
2024:OUT 80        ；输出保存在 R0 低位字节的字符
2025:DEC R2        ；输出字符个数减 1
2026:JRZ 202E      ；判 10 个字符输出完否,已完，则转到程序结束处
2027:PUSH R0       ；未完，保存 R0 的值到堆栈中
2028:IN  81        ；查询接口状态，判字符串行输出完成否,
2029:SHR R0        ；
202A:JRNC 2028     ；未完成，则循环等待
```

```
202B:POP R0          ；已完成，准备输出下一字符并从堆栈恢复 R0 的值
202C:INC R0          ；得到下一个要输出的字符
202D:JR  2024        ；转去输出字符
202E:RET
202F：↙
```

该程序的执行码放在 2020H 起始的连续内存区中。若送入源码的过程中有错，系统会进行提示，等待重新输入正确汇编语句。在输入过程中，在应输入语句的位置直接打回车则结束输入过程。

〈2〉用“G”命令运行程序

在命令行提示符状态下输入：

G 2020↙

执行结果为：

0123456789

思考题：若把 IN 81，SHR R0，JRNC 2028 三个语句换成 4 个 MVRR R0，R0 语句，该程序执行过程会出现什么现象？试分析并实际执行一次。

提示：该程序改变这三条语句后，若用 T 命令单条执行，会依次显示 0~9 十个数字。若用 G 命令运行程序，程序执行速度快，端口输出速度慢，这样就会跳跃输出。

在命令行提示符状态下输 G 2020，屏幕显示 09。

类似的，若要求在终端屏幕上输出’A’到’Z’共 26 个英文字母，应如何修改例 1 中给出的程序？请验证之。

参考答案：

在命令行提示符状态下输入：

A 2100↙

屏幕将显示：2100：

从地址 2100H 开始输入下列程序：

```
(2100) MVRD R2，001A   ；循环次数为 26
       MVRD R0，0041   ；字符“A”的值
(2104) OUT 80          ；输出保存在 R0 低位字节的字符
       DEC R2          ；输出字符个数减 1
       JRZ 210E        ；判 26 个字符输出完否,已完，则转移到程序结束处
       PUSH R0         ；未完，保存 R0 的值到堆栈中
(2108) IN 81           ；查询接口状态，判字符串行输出完成否
       SHR R0
       JRNC 2108       ；未完成，则循环等待
       POP R0          ；已完成，准备输出下一字符，从堆栈恢复 R0 的值
       INC R0          ；得到下一个要输出的字符
       JR 2104         ；转去输出字符
(210E) RET
```

用 G 命令执行该程序，屏幕上显示“A”~“Z”26 个英文字母.

例 3:从键盘上连续打入多个属于‘0’到‘9’的数字符并在屏幕上显示,遇到非数字字

符结束输入过程。

<1> 在命令行提示符状态下输入：

A 2040↙

屏幕将显示：

2040：

从地址 2040H 开始输入下列程序:

```
(2040)  MVRD R2,0030        ;  用于判数字符的下界值
        MVRD R3,0039        ;  用于判数字符的上界值
(2044)  IN 81               ;  判键盘上是否按了一个键,
        SHR R0              ;  即串行口是否有了输入的字符
        SHR R0
        JRNC 2044           ;  没有输入则循环测试
        IN 80               ;  输入字符到 R0
        MVRD R1,00FF
        AND R0,R1           ; 清零 R0 的高位字节内容
        CMP R0,R2           ;  判输入字符≥字符'0'否
        JRNC 2053           ;  为否, 则转到程序结束处
        CMP R3,R0           ;  判输入字符≤字符'9'否
        JRNC 2053           ;  为否, 则转到程序结束处
        OUT 80              ;  输出刚输入的数字符
        JMPA 2044                ; 转去程序前边 2044 处等待输入下一个字符
(2053)  RET
```

〈2〉 在命令行提示符状态下输入：

G 2040↙

光标闪烁等待键盘输入，若输入 0-9 十个数字符，则在屏幕上回显；若输入非数字符，则屏幕不再显示该字符，出现命令提示符，等待新命令。

思考题, 本程序中为什么不必判别串行口输出完成否? 设计打入'A'～'Z'和'0'～'9'的程序, 遇到其它字符结束输入过程。

例子 4: 计算 1 到 10 的累加和。

〈1〉在命令行提示符状态下输入：

A 2060↙

屏幕将显示：

2060：

从地址 2060H 开始输入下列程序：

```
(2060) MVRD R1,0000         ;  置累加和的初值为 0
       MVRD R2,000A         ;  最大的加数
       MVRD R3,0000
(2066) INC R3               ;  得到下一个参加累加的数
```

```
        ADD R1,R3          ;  累加计算
        CMP R3,R2          ;  判是否累加完
        JRNZ 2066          ;  未完，开始下一轮累加
        RET
```

〈2〉 在命令行提示符状态下输入：

G 2060↙

运行过后，可以用 R 命令观察累加器的内容。R1 的内容为累加和。

结果为：R1=0037 R2=000A R3=000A

例子 5:设计一个有读写内存和子程序调用指令的程序，功能是读出内存中的字符，将其显示到显示器的屏幕上，转换为小写字母后再写回存储器原存储区域。

〈1〉 将被显示的 6 个字符‘A’～‘F’送入到内存 20F0H 开始的存储区域中。

在命令行提示符状态下输入：

E 20F0↙

屏幕将显示：

20F0　　内存单元原值：

按下列格式输入：

20F0　内存原值：0041　内存原值：0042　内存原值：0043

　　内存原值：0044　内存原值：0045　内存原值：0046↙

在命令行提示符状态下输入：

从地址 2080H 开始输入下列程序：

```
(2080) MVRD R3,0006          ; 指定被读数据的个数
       MVRD R2,20F0          ; 指定被读、写数据内存区首地址
(2084) LDRR R0,[R2]          ; 读内存中的一个字符到 R0 寄存器
       CALA 2100             ; 指定子程序地址为 2100,调用子程序，完成显示、
                               转换并写回的功能
       DEC R3                ; 检查输出的字符个数
       JRZ 208B              ; 完成输出则结束程序的执行过程
       INC R2                ; 未完成，修改内存地址
       JR 2084               ; 转移到程序的 2086 处，循环执行规定的处理
(208B) RET
```

从地址 2100H 开始输入下列程序：

```
(2100) OUT 80                ; 输出保存在 R0 寄存器中的字符
       MVRD R1，0020
       ADD R0，R1            ; 将保存在 R0 中的大写字母转换为小写字母
       STRR [R2],R0          ; 写 R0 中的字符到内存，地址同 LOD 所用的地址
(2105) IN 81                 ; 测试串行接口是否完成输出过程
       SHR R0
       JRNC 2105             ; 未完成输出过程则循环测试
```

```
        RET                 ；结束子程序执行过程，返回主程序
```

〈2〉在命令行提示符状态下输入：

G 2080↙

屏幕显示运行结果为：

ABCDEF

〈3〉在命令行提示符状态下输入：

D 20F0↙

20F0H~20F5H 内存单元的内容为：

0061 0062 0063 0064 0065 0066

例子 6：设计一个程序在显示器屏幕上循环显示 95 个（包括空格字符）可打印字符。

〈1〉在命令行提示符状态下输入：

A 20A0↙

屏幕将显示：

20A0：

从地址 20A0H 开始输入下列程序：

```
A  20A0              ；从内存的 20A0 单元开始建立用户的第一个程序
20A0:  MVRD R1,7E    ；向寄存器传送立即数
20A2:  MVRD R0,20    ；
20A4:  OUT 80        ；通过串行接口输出 R0 低位字节内容到显示器屏幕
20A5:  PUSH R0       ；保存 R0 寄存器的内容到堆栈中
20A6:  IN 81         ；读串行接口的状态寄存器的内容
20A7:  SHR R0        ；R0 寄存器的内容右移一位，最低位的值移入标志位 C
20A8:  JRNC 20A6     ；条件转移指令，当标志位 C 不是 1 时就转到 20A6 地址
20A9:  POP R0        ；从堆栈中恢复 R0 寄存器的原内容
20AA:  CMP R0,R1     ；比较两个寄存器的内容是否相同，相同则标志位 Z=1
20AB:  JRZ 20A0      ；条件转移指令，当标志位 Z 为 1 时转到 200A0 地址
20AC:  INC R0        ；把 R0 寄存器的内容增加 1
20AE:  JR 20A4       ；无条件转移指令，一定转移到 20A4 地址
20AF:  RET           ；子程序返回指令,程序结束
```

〈2〉在命令行提示符状态下输入：

G 20A0↙

运行过后，可以观察到显示器上会显示出所有可打印的字符。

上述例子，都是用监控程序的 A 命令完成输入源汇编程序的。在涉及到汇编语句标号的地方，不能用符号表示，只能在指令中使用绝对地址。使用内存中的数据，也由程序员给出数据在内存中的绝对地址。显而易见，对这样的极短小程序矛盾并不突出，但很容易想到，对很大的程序，一定会有较大的困难。

在用 A 命令输入汇编源语句的过程中，有一定用机经验的人，常常抱怨 A 命令中未提供适当的编辑功能，这并不是设计者的疏漏，因为我们并不准备在这种操作方式下支持设计较长的程序，这种工作应转到提供了交叉汇编程序的 PC 机上去完成。相反的情况是，输入上述一些小程序，用监控程序的 A 命令完成，往往比用交叉汇编完成更简捷。

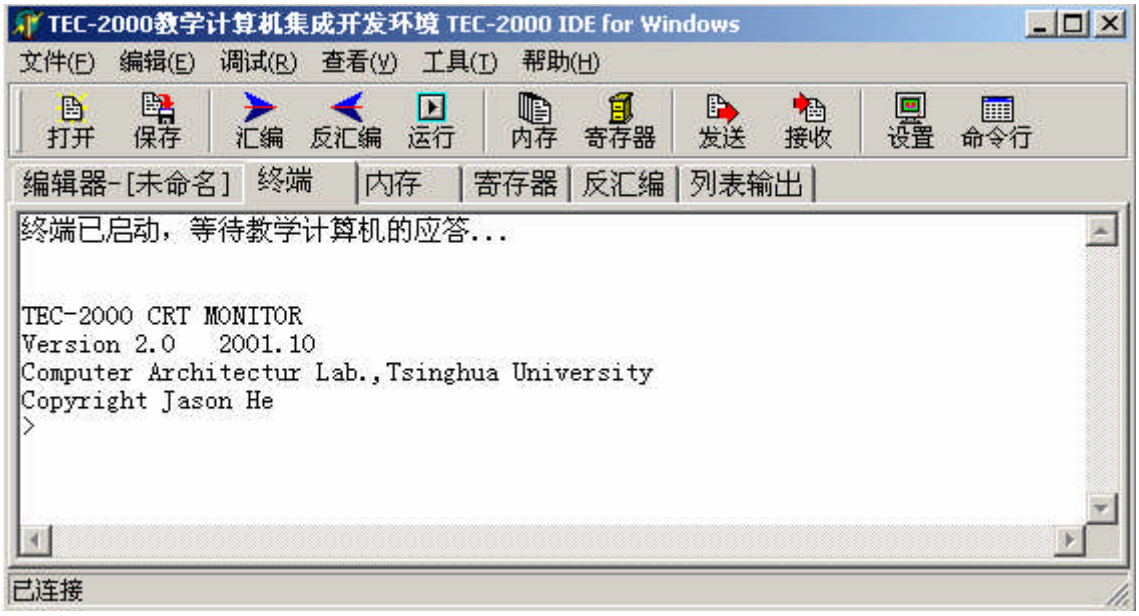
七．教学计算机的 WINDOWS 界面的集成开发环境

上述的实验除了在 PCEC 仿真终端的界面下完成外，也可以在 WINDOWS 界面的集成开发环境下完成。用户可以将配套光盘中的 TEC2KIDE 的文件夹复制到硬盘建立的 TEC-XP16 的文件夹中。然后选中文件夹中的可执行文件 TEC2KIDE 单击鼠标右键，在弹出的菜单中选择“发送到”项中的“桌面快捷方式”一项，建立一个放到桌面上的快捷方式。

1．双击桌面上的 TEC2KIDE 图标，会出现如下的软件界面：

2．选中菜单栏中的“终端”选项，打开连接好的教学机的电源，会出现如下通讯界面：

3．按一次“RESET”按键，再按一次“START”按键，出现如图所示界面：



4．按照第六个实验项目中给出的示例输入程序，并执行，可以观察到同样的实验结果。

5．建立一个汇编的源文件，并进行编译，下载。

选中菜单栏中的“编辑器”选项，可以建立汇编的源文件，也可以打开硬盘上已经建立好的源文件。选中菜单栏中的“汇编”选项，对源文件进行汇编，然后可以选择“发送”选项将文件传送到教学机上运行。

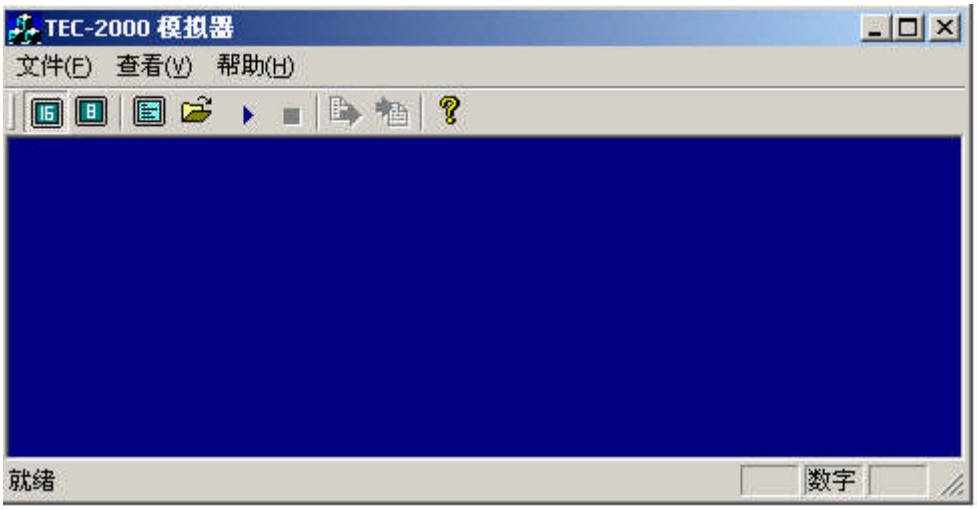
6．观察内存和寄存器中的内容。

八．软件模拟器

为了让学生更好的理解机器的指令系统，我们设计了软件模拟器，联机通讯所能实现的功能在软件模拟器上也一样能实现，这个软件可以让学生拷贝回去，在自己的机器上运行汇编语言的程序，能更好的理解机器的监控程序的功能以及指令系统，对学生的学习有很好的促进作用。

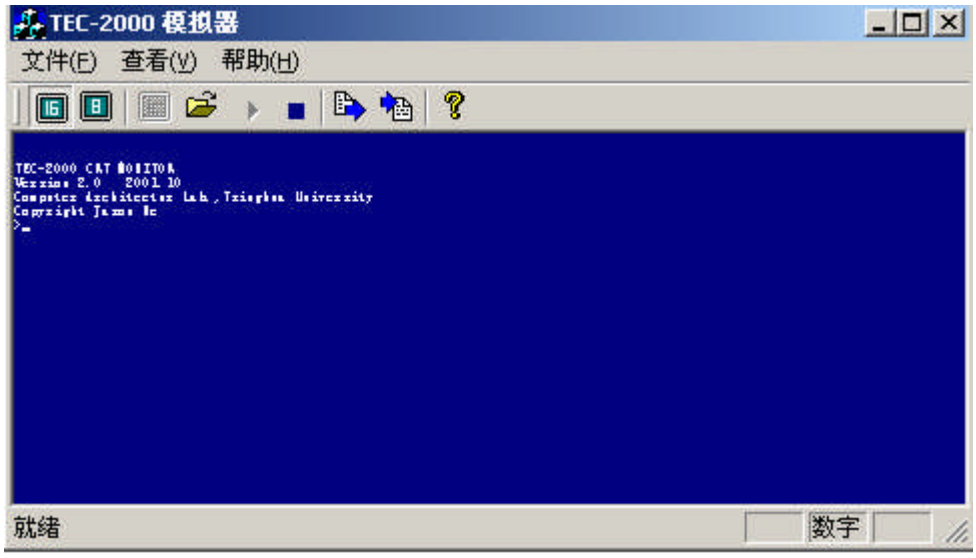
用户可以将配套光盘中的 TEC2KSIM 的文件夹复制到硬盘建立的 TEC-XP16 的文件夹中。然后选中该文件夹中的可执行文件 TEC2KSIM 单击鼠标右键，在弹出的菜单中选择“发送到”项中的“桌面快捷方式”一项，建立一个放到桌面上的快捷方式。

1． 双击桌面上的 TEC2KIDE 图标，会出现如下的软件界面：



2． 单击标有“16”的图标，选中 16 位机的运行模式；再单击第三项“启动监控”的图标，

会出现如下的界面：



3. 在这个界面中可以使用监控程序支持的 DEBUG 命令，输入并且执行程序，也可以观察修改地址单元内容和寄存器的内容。

2.2 脱机运算器实验

实验目的

深入了解 AM2901 运算器的功能与具体用法，4 片 AM2901 的级连方式，深化运算器部件的组成、设计、控制与使用等诸项知识。

实验说明

脱机运算器实验，是指让运算器从教学计算机整机中脱离出来，此时，它的全部控制与操作均需通过两个 12 位的微型开关来完成，这就谈不上执行指令，只能通过开关、按键控制教学机的运算器完成指定的运算功能，并通过指示灯观察运算结果。

下面先把前边讲过的、与该实验直接有关的结论性内容汇总如下。

一. 12 位微型开关的具体控制功能分配如下：

A 口、B 口地址：送给 AM2901 器件用于选择**源**与**目的**操作数的寄存器编号；

I8-I0：选择操作数来源、运算操作功能、选择操作数处理结果和运算器输出内容的 3 组 3 位的控制码；

SCi、SSH 和 SST：用于确定运算器最低位的进位输入、移位信号的入/出和怎样处理 AM2901 产生的状态标志位的结果。

二. 开关位置说明：

做脱机运算器实验时，要用到提供 24 位控制信号的微动开关和提供 16 位数据的拨动开关。微动开关是红色的，一共有三个，一个微动开关可以提供 12 位的控制信号，三个开关分

别标有 SW1 micro switch 、SW2 micro switch 和 SW3 micro switch ，他们对应的控制信号见下表；数据开关是黑色的，左边的标有 SWH 的是高 8 位，右边的标有 SWL 的是低 8 位。

微动开关与控制信号对应关系见表(由左到右)：

SW1 Micro switch				SW2 Micro switch			SW3 Micro switch		
T3-T0	REQ/MIO/WE	I2-I0	I8-I7	I6-I3	B PORT	A PORT	SST SSH SCI	DC2	DC1

三． 开关检测

红色微动开关是该实验系统使用寿命最短的器件，开关好坏的检测方法比较简单，用户将五个控制机器工作方式的开关置于“1XX00”,从左面起第二个和第三个的开关处于任意位置，然后将三个微动开关上的小纽子（节拍除外）依次置为 1（开关拨到上方为 1），看对应的指示灯是否亮，然后再将其置为 0，相应的指示灯会灭，如果有一个或数个指示灯不能正常点亮或者灭，则一般是开关出了问题。

实验步骤

1．将教学机左下方的 6 个拨动开关置为 **1XX000**（单步、16 位、脱机、MACH）；先按一下“RESET”按键，再按一下“START”按键，进行初始化。

2．接下来，按下表所列的操作在机器上进行运算器脱机实验，将结果填入表中：其中 D1 取为 0101H，D2 取为 1010H；通过两个 12 位的红色微型开关向运算器提供控制信号，通过 16 位数据开关向运算器提供数据，通过指示灯观察运算结果及状态标志。

运算	I8-I0	SST	SSH	SCi	B	A	压 START 前		压 START 后	
							ALU 输出	CZVS	ALU 输出	CZVS
***D1+0->R0**	011**000**111	001	00	00	0000	不用	0101	随机	0101	0000
***D2+0->R1**	011**000**111	001	00	00	0001	不用	1010	0000	1010	0000
R0+R1->R0	011**000**001	001	00	00	0000	0001	1111	0000	2121	0000
R0-R1->R0	011**001**001	001	00	01	0000	0001	0101	0000	F0F1	1000
R1-R0->R1	011**001**001	001	00	01	0001	0000	0F0F	1000	0E0E	1000
R0∨R1->R0	011**011**001	001	00	00	0001	0000	0F0F	1000	0F0F	1000
R0∧R1->R0	011**100**001	001	00	00	0000	0001	0101	1000	0101	1000
R0⊻R1->R0	011**110**001	001	00	00	0000	0001	0E0E	1000	0101	1000
¬ (R0⊻R1)->R0	011**111**001	001	00	00	0000	0001	FEFE	1000	0E0E	1001
2*R0->R0	111**000**011	001	00	00	0000	不用	FEFE	1001	FDFC	0001
R0/2->R0	101**000**011	001	00	00	0000	不用	FDFC	0001	7EFE	0001

注：用*标记的运算，表示 D1、D2 的数据是由拨动开关 SW 给出的，开关给的是二进制的信号，注意二进制和十六进制间的转换。

按“START”按键之前，ALU 输出的是计算结果，参照 ALU 的操作周期的时序可知 A、B 口数据锁存是在时钟的下降沿，通用寄存器的接收是在低电平，所以要想寄存器接收 ALU 的计算结果必须按一次“START”按键。

实验要求

1．实验之前认真预习，写出预习报告，包括操作步骤，实验过程所用数据和运行结果等，否则实验效率会很低，所学到的内容也会大受影响；

2．实验过程当中，要仔细进行，防止损坏设备，分析可能遇到的各种现象，判断结果是否正确，记录运行结果；

3．实验之后，认真写出实验报告，包括对遇到的各种现象的分析，实验步骤和实验结果，自己在这次实验的心得体会与收获。

2.3 组合逻辑控制器部件教学实验

实验目的

通过看懂教学计算机中已经设计好并正常运行的几条典型指令（例如，ADD、SHR、OUT、MVRD、JRC、RET、CALA 等指令）的功能、格式和执行流程，然后自己设计几条指令的功能、格式和执行流程，并在教学计算机上实现、调试正确。其最终要达到的目的是：

1. 深入理解计算机控制器的功能、组成知识；
2. 深入地学习计算机各类典型指令的执行流程；
3. 对指令格式、寻址方式、指令系统、指令分类等建立具体的总体概念；
4. 学习组合逻辑控制器的设计过程和相关技术。

实验说明

控制器设计是学习计算机总体组成和设计的最重要的部分。要在 TEC-XP＋教学计算机完成这项实验，必须比较清楚地懂得：

1. TEC-XP＋教学机的组合逻辑控制器主要由 MACH 器件组成；
2. TEC-XP＋教学机上已实现了 29 条基本指令的控制信号由 MACH 给出的
3. 应了解监控程序的 A 命令只支持基本指令，扩展指令应用 E 命令将指令代码写入到相应的存储单元中；不能用 T、P 命令单步调试扩展指令，只能用 G 命令执行有扩展指令的程序。
4. 要明白 TEC-XP＋教学机支持的指令格式及指令执行流程分组情况；理解 TEC-XP＋教学机中已经设计好并正常运行的各类指令的功能、格式和执行流程，也包括控制器设计与实现中的具体线路和控制信号的组成。
5. 要明确自己要实现的指令格式、功能、执行流程设计中必须遵从的约束条件。

   为了完成自己设计几条指令的功能、格式和执行流程，并在教学计算机上实现、调试正确的实验内容，具体过程包括：

   1) 确定指令格式和功能，要受教学机已有硬件的约束，应尽量与已实现指令的格式和分类办法保持一致；
   2) 划分指令执行步骤并设计每一步的执行功能，设计节拍状态的取值，应参照已实现指令的处理办法来完成，特别要注意的是，读取指令的节拍只能用原来已实现的，其他节拍的节拍状态也应尽可能地与原用节拍的状态保持一致和相近；
   3) 在指令流程表中填写每一个控制信号的状态值，基本上是个查表填数的过程，应该特别仔细，并有意识地体会这些信号的控制作用；
   4) 在提供的 MACH 源文件的基础上写出每个控制信号的完整逻辑表达式，可能和必要的话，进行一点逻辑化简；

5) 对生成的源文件进行编译，将编译适配生成的熔丝图文件.JED 文件下载到 MACH 器件中去；

6) 写一个包含你设计的指令的程序，通过运行该程序检查执行结果的正确性，来初步判断你的设计是否正确；如果有问题，通过几种办法查出错误并改正，（比如手动置指令，单步调试每个节拍对应的控制信号）继续调试，直到完全正确。

实验内容

1. 完成控制器部件的教学实验，主要内容是由学生自己设计几条指令的的功能、格式和执行流程，并在教学计算机上实现、调试正确。
2. 首先是看懂 TEC-XP＋教学计算机的功能部件组成和线路逻辑关系，然后分析教学计算机中已经设计好并正常运行的几条典型指令（例如，ADD、SHR、OUT、MVRD、JRC、CALA、RET 等指令）的功能、格式和执行流程。
3. 设计几条指令的功能、格式和执行流程，并在教学计算机上实现、调试正确。例如 ADC、JRS、JRNS、LDRA、STOR、JMPR 等指令，可以从《TEC-XP＋教学计算机系统技术说明与实验指导》第二章给出的 19 条扩展指令中任意选择，当然也可以设计与实现其它的指令，包括原来已经实现的基本指令（要变换为另外一个指令操作码）或自己确定的指令。在原来提供的 MACH 程序的基础上按照 ABEL 语言的要求添加新指令的控制信号，编译产生.JED 文件并下载到 MACH 芯片里。软件的使用和下载参加附录。
4. 单条运行指令，查看指令的功能、格式和执行流程。
   先将教学机左下方的 6 个拨动开关置为 111100，再按一下“RESET”按键，然后通过 16 位的数据开关（SWH、SWL）置入指令，按“START”按键单步送脉冲，通过指示灯观察控制信号的变化。
5. 用监控程序的 A、E（扩展指令必须用 E 命令置入）命令编写一段小程序，观察运行结果。
   实验时将教学机左下方的 5 个拨动开关置为 001100，运行编写的小程序。观察终端显示的结果，检验设计的指令是否正确。若与预定结果不符，可查看指令的功能、格式、执行、流程设计的是否正确。

实验要求

1. 实验之前，应认真准备，写出实验步骤和具体设计内容，否则实验效率会特别低，一次实验时间根本无法完成实验任务，即使基本做对了，也很难说学懂了些什么重要教学内容；
2. 应在实验前掌握所有控制信号的作用，在脱机运算器实验中，已给出了与运算器有关的控制信号的作用，16 位机组合逻辑控制器用到的控制信号的功能表参见《技术说明和实验指导》。
3. 实验过程中，应认真进行实验操作，既不要因为粗心造成短路等事故而损坏设备，又要仔细思考实验有关内容，提高学习的主动性和创造性，把自己想不太明白的问题通过实验理解清楚，争取最好的实验效果，力求达到教学实验的主要目的；
4. 实验之后，应认真思考总结，写出实验报告，包括实验步骤和具体实验结果，遇到的主要问题和分析与解决问题的思路。大家应该认识到，遇到一些问题是好事情，通过分析与解决这些问题，才提高了自己的工作能力，学习到更多的知识。还未理解清楚，但实验结果正确了就匆忙结束实验，并没有达到教学实验的目的。实验报告中，还应写出自己的学习

心得和切身体会，也可以对教学实验提出新的建议等。实验报告要交给教师评阅并给出实验成绩。

实验步骤

1. 接通教学机电源；
2. 将教学机左下方的 6 个拨动开关置为 111100（单步、手动置指令、组合、16 位、联机、MACH）；
3. 按一下“RESET”按键；
4. 通过 16 位的数据开关 SWH、SWL 置入 16 位的指令操作码；
5. 在单步方式下，通过指示灯观察**各类基本指令的节拍**。

1）选择基本指令的 A 组指令中的 **ADD** 指令，观察其节拍流程：

〈1〉 置拨动开关 SW=00000000 00000001；（表示指令 ADD R0，R1 ）

〈2〉 按 RESET 按键；节拍指示灯 T4~T0 显示 0 1000；(本拍在第 1 次复位后才会执行)

〈3〉 按 START 按键；节拍指示灯 T4~T0 显示 0 0000；(以上两拍,为公共节拍，在手动置指令方式下无意义)

〈4〉 按 START 按键；节拍指示灯 T4~T0 显示 0 0010；(本拍也是公共节拍，将指令编码写入指令寄存器 IRH、IRL)

〈5〉 按 START 按键；节拍指示灯 T4~T0 显示 0 0011；(本拍执行 ADD 指令，R0←R0+R1 操作)

可以看到，A 组指令（包括 ADD、SUB、CMP、AND、XOR、SHR、SHL、INC、DEC、TEST、OR、MVRR、JR、JRC、JRNC、JRZ、JRNZ）的执行除公共节拍外，只需一步完成。

2）选择基本指令的 B 组指令中的 **PUSH** 指令，观察其节拍流程：

〈1〉置拨动开关 SW=10000101 00000000；（表示指令 PUSH ）

〈2〉按 RESET 按键；节拍指示灯 T4~T0 显示 0 1000；(本拍在第 1 次复位后才会执行)

〈3〉按 START 按键；节拍指示灯 T4~T0 显示 0 0000；(以上两拍,为公共节拍，在手动置指令方式下无意义)

〈4〉按 START 按键；节拍指示灯 T4~T0 显示 0 0010；；(本拍也是公共节拍，将指令编码写入指令寄存器 IRH、IRL)

〈5〉按 START 按键；节拍指示灯 T4~T0 显示 0 0110（本拍执行 PUSH 指令的第一步，修改地址寄存器和堆栈的值，即 AR,SP←SP-1，使其指向堆栈空间）

〈6〉按 START 按键；节拍指示灯 T4~T0 显示 0 0100；（本拍执行 PUSH 指令的第二步，MEM←SR）

可以看到，B 组指令（包括 JMPA、LDRR、IN、STRR、PSHF、PUSH、OUT、POP、MVRD、POP、RET）的执行除公共节拍外,需两步完成。

3） 选择基本指令的 D 组指令中的 **CALA** 指令，观察其节拍流程：

〈1〉 置拨动开关 SW=11001110 00000000；（表示指令 CALA）

〈2〉 按 RESET 按键；节拍指示灯 T4~T0 显示 0 1000；(本拍在第 1 次复位后才会执行)

〈3〉 按 START 按键；节拍指示灯 T4~T0 显示 0 0000；(以上两拍,为公共节拍，在手动置指令方式下无意义)

〈4〉按 START 按键；节拍指示灯 T4~T0 显示 0 0010；（本拍也是公共节拍，将指令编码写入指令寄存器 IRH、IRL）

〈5〉按 START 按键；节拍指示灯 T4~T0 显示 0 0110；（本拍 PC→AR,PC+1→PC）

〈6〉按 START 按键；节拍指示灯 T4~T0 显示 0 0100；（本拍(AR)→Q）

〈7〉按 START 按键；节拍指示灯 T4~T0 显示 0 0111；（本拍 SP-1→SP、AR）

〈8〉按 START 按键；节拍指示灯 T4~T0 显示 0 0101；（本拍 PC→MEM，Q→PC）

可以看到，D 组指令 CALA 除公共节拍外，需四步完成；

6．单步方式下，通过指示灯观察**各类基本指令的控制信号**。

1）选择基本指令的 A 组指令中的 **SHR** 指令，观察其执行过程中控制信号的变化，分析其作用。

〈1〉置拨动开关 SW=00001011 00010000；(表示指令 SHR R1)

〈2〉先按“RESET”按键；再连续按“START”按键，观察每一步的节拍及控制信号如下表：

节拍	指令	编码	/MIO	REQ	/WE	A	B	Sci	SSH	I8-6	I5-3	I2-0	SST	DC1	DC2
1000			1	0	0	0101	0101	01	0	011	001	001	000	000	111
0000			1	0	0	0101	0101	01	0	010	000	011	000	000	011
0010			0	0	1	0000	0000	00	0	001	000	000	000	000	001
0011	SHR	0000 1011	1	0	0	0000	0001	00	0	101	000	011	101	000	000

2）选择基本指令的 B 组指令中的 **JMPA** 指令，观察其执行过程中控制信号的变化，分析其作用。

〈1〉置拨动开关 SW=10000000 00000000；(表示指令 JMPA)

〈2〉先按“RESET”按键；再连续按“START”按键，观察每一步的节拍及控制信号如下表：

节拍	指令	编码	/MIO	REQ	/WE	A	B	Sci	SSH	I8-6	I5-3	I2-0	SST	DC1	DC2
1000			1	0	0	0101	0101	01	0	011	001	001	000	000	111
0000			1	0	0	0101	0101	01	0	010	000	011	000	000	011
0010			0	0	1	0000	0000	00	0	001	000	000	000	000	001
0110	JMPA	1000 0000	1	0	0	0101	0101	01	0	010	000	011	000	000	011
0100	JMPA	1000 0000	0	0	1	0000	0101	00	0	011	000	111	000	000	000

3）选择基本指令的 D 组指令中的 **CALA** 指令，观察其执行过程中控制信号的变化，分析其作用。

〈1〉 置拨动开关 SW=11001110 00000000；（表示指令 CALA）

〈2〉 先按“RESET”按键；再连续按“START”按键，观察每一步的节拍及控制信号如下表：

节拍	指令	编码	/MIO	REQ	/WE	A	B	Sci	SSH	I8-6	I5-3	I2-0	SST	DC1	DC2
1000			1	0	0	0101	0101	01	0	011	001	001	000	000	111
0000			1	0	0	0101	0101	01	0	010	000	011	000	000	011
0010			0	0	1	0000	0000	00	0	001	000	000	000	000	001

0110	CALA	1100 1110	1	0	0	0101	0101	01	0	010	000	011	000	000	011
0100	CALA	1100 1110	0	0	1	0000	0000	00	0	000	000	111	000	000	000
0111	CALA	1100 1110	1	0	0	0000	0100	00	0	011	001	011	000	000	011
0101	CALA	1100 1110	0	0	0	0101	0101	00	0	010	000	010	000	001	000

7. 在以上几步实验的基础上，选择几条扩展指令，设计出扩展指令的节拍和每拍对应的控制信号。（节拍的设计参加节拍的流程图，扩展指令的节拍，在出厂时的 TIMING GAL 中已实现，学生可以不用设计，只需看懂节拍 GAL 的逻辑表达式即可,但其控制信号需用户来扩展实现，这一步，只是来观察扩展指令的节拍。）

单步方式下，通过指示灯**观察各类扩展指令的节拍**

1) 选择扩展指令的 A 组指令中的 **RCR** 指令，观察其节拍流程：

〈1〉置拨动开关 SW=00101011 00010000；（表示指令 RCR R1）

〈2〉按 RESET 按键；节拍指示灯 T4~T0 显示 0 1000；(本拍在第 1 次复位后才会执行)

〈3〉按 START 按键；节拍指示灯 T4~T0 显示 0 0000；(以上两拍为公共节拍，在手动置指令方式下无意义)

〈4〉按 START 按键；节拍指示灯 T4~T0 显示 0 0010；（本拍也是公共节拍，将指令编码写入指令寄存器 IRH、IRL）

〈5〉按 START 按键；节拍指示灯 T4~T0 显示 0 0011；（本拍完成循环右移操作，RCR DR）

可以看到，A 组扩展指令（包括 ADC、SBB、RCL、RCR、NOT、JMPR、ASR、JRS、JRNS、CLC、STC、EI、DI）除公共节拍外，只需一步完成。

2) 选择扩展指令的 C 组指令中的 **LDRA** 指令，观察其节拍流程：

〈1〉 置拨动开关 SW=11100100 00000000；（表示指令 LDRA）

〈2〉 按 RESET 按键；节拍指示灯 T4~T0 显示 0 1000；(本拍在第 1 次复位后才会执行)

〈3〉 按 START 按键；节拍指示灯 T4~T0 显示 0 0000；(以上两拍为公共节拍，在手动置指令方式下无意义)

〈4〉 按 START 按键；节拍指示灯 T4~T0 显示 0 0010；（本拍也是公共节拍，将指令编码写入指令寄存器 IRH、IRL）

〈5〉 按 START 按键；节拍指示灯 T4~T0 显示 0 0110；(本拍完成操作 PC→AR，PC+1→PC)

〈6〉 按 START 按键；节拍指示灯 T4~T0 显示 0 0111；（本拍完成操作 MEM→AR）

〈7〉 按 START 按键；节拍指示灯 T4~T0 显示 0 0101；（本拍完成操作 MEM→DR）

可以看到，C 组扩展指令（包括 CALR、LDRA、LDOR、STOR、STAR）除公共节拍外，需三步完成。

8. 设计几条扩展指令的控制信号如下表：

〈1〉 选择扩展指令 ADC、STC、JRS、LDRX、STRX 和 JMPR，其节拍和设计的控制信号为：

节拍	指令	编码	/MIO	REQ	/WE	A	B	Sci	SSH	I8-6	I5-3	I2-0	SST	DC1	DC2
1000			1	0	0	0101	0101	01	0	011	001	001	000	000	111
0000			1	0	0	0101	0101	01	0	010	000	011	000	000	011
0010			0	0	1	0000	0000	00	0	001	000	000	000	000	001
0011	ADC	0010 0000	1	0	0	SR	DR	10	0	011	000	001	001	000	000
	JRS	0110 0100	1	0	0	0101	0101	00	0	0S1	000	101	000	010	000
	STC	0110 1101	1	0	0	0000	0000	00	0	001	000	000	100	000	000
	JMPR	0110 0000	1	0	0	SR	0101	00	0	011	000	100	000	000	000
0110	LDRX	1110 0101	1	0	0	0101	0101	01	0	010	000	011	000	000	011
	STRX	1110 0110	1	0	0	0101	0101	01	0	010	000	011	000	000	011
0111	LDRX	1110 0101	0	0	1	SR	0000	00	0	001	000	101	000	000	011
	STRX	1110 0110	0	0	1	SR	0000	00	0	001	000	101	000	000	011
0101	LDRX	1110 0101	0	0	1	0000	DR	00	0	011	000	111	000	000	000
	STRX	1110 0110	0	0	0	0000	DR	00	0	001	000	011	000	001	000

〈2〉 根据设计的控制信号的表格用 ABEL 语言编写 MACH 的逻辑表达式，老师可以参考提供的组合逻辑全指令的 MACH 程序的逻辑表达式，具体内容参见用户光盘；

〈3〉 将编译好的程序 MACHC.JED 下载到 MACH 芯片内（编译和下载使用 ISP LEVER 软件，具体的操作步骤参见《TEC-XP＋教学计算机技术说明和实验指导》附录）。

9. 单步方式下，通过指示灯观察上面扩展的几条**扩展指令的控制信号**是否与设计的一致。

1） 观察 A 组指令中的 **ADC** 指令：

〈1〉 置拨动开关 SW=00100000 00010000；

〈2〉 先按“RESET”按键；再连续按“START”按键，观察每一步的节拍及控制信号如下表：

节拍	指令	编码	/MIO	REQ	/WE	A	B	Sci	SSH	I8-6	I5-3	I2-0	SST	DC1	DC2
1000			1	0	0	0101	0101	01	0	011	001	001	000	000	111
0000			1	0	0	0101	0101	01	0	010	000	011	000	000	011
0010			0	0	1	0000	0000	00	0	001	000	000	000	000	001
0011	ADC	0010 0000	1	0	0	SR	DR	10	0	011	000	001	001	000	000

2）观察 A 组指令中的 JRS 指令：

〈1〉 置拨动开关 SW=01100100 00000000；

〈2〉 先按“RESET”按键；再连续按“START”按键，观察每一步的节拍及控制信号如下表。

节拍	指令	编码	/MIO	REQ	/WE	A	B	Sci	SSH	I8-6	I5-3	I2-0	SST	DC1	DC2
1000			1	0	0	0101	0101	01	0	011	001	001	000	000	111
0000			1	0	0	0101	0101	01	0	010	000	011	000	000	011
0010			0	0	1	0000	0000	00	0	001	000	000	000	000	001
0011	JRS	0110 0100	1	0	0	0101	0101	00	0	0S1	000	101	000	010	000

3） 观察 A 组指令中的 STC 指令：

〈1〉 置拨动开关 SW=01101101 00000000；

〈2〉 先按“RESET”按键；再连续按“START”按键，观察每一步的节拍及控制信号如下表。

节拍	指令	编码	/MIO	REQ	/WE	A	B	Sci	SSH	I8-6	I5-3	I2-0	SST	DC1	DC2
1000			1	0	0	0101	0101	01	0	011	001	001	000	000	111
0000			1	0	0	0101	0101	01	0	010	000	011	000	000	011
0010			0	0	1	0000	0000	00	0	001	000	000	000	000	001
0011	STC	0110 1101	1	0	0	0000	0000	00	0	001	000	000	100	000	000

4）观察 A 组指令中的 JMPR 指令：

〈1〉 置拨动开关 SW=01100000 00000001；

〈2〉 先按“RESET”按键；再连续按“START”按键，观察每一步的节拍及控制信号如下表。

节拍	指令	编码	/MIO	REQ	/WE	A	B	Sci	SSH	I8-6	I5-3	I2-0	SST	DC1	DC2
1000			1	0	0	0101	0101	01	0	011	001	001	000	000	111
0000			1	0	0	0101	0101	01	0	010	000	011	000	000	011
0010			0	0	1	0000	0000	00	0	001	000	000	000	000	001
0011	JMPR	0110 0000	1	0	0	0001	0101	00	0	011	000	100	000	000	000

5）观察 C 组指令中的 LDRX 指令：

〈1〉 置拨动开关 SW=11100101 00000000；

〈2〉 先按“RESET”按键；再连续按“START”按键，观察每一步的节拍及控制信号。

节拍	指令	编码	/MIO	REQ	/WE	A	B	Sci	SSH	I8-6	I5-3	I2-0	SST	DC1	DC2
1000			1	0	0	0101	0101	01	0	011	001	001	000	000	111
0000			1	0	0	0101	0101	01	0	010	000	011	000	000	011
0010			0	0	1	0000	0000	00	0	001	000	000	000	000	001
0110	LDRX	1110 0101	1	0	0	0101	0101	01	0	010	000	011	000	000	011
0111	LDRX	1110 0101	0	0	1	SR	0000	00	0	001	000	101	000	000	011
0101	LDRX	1110 0101	0	0	1	0000	DR	00	0	011	000	111	000	000	000

6）观察 C 组指令中的 STRX 指令：

〈1〉 置拨动开关 SW=11100110 00000000；

〈2〉 先按“RESET”按键；再连续按“START”按键，观察每一步的节拍及控制信号。

节拍	指令	编码	/MIO	REQ	/WE	A	B	Sci	SSH	I8-6	I5-3	I2-0	SST	DC1	DC2
1000			1	0	0	0101	0101	01	0	011	001	001	000	000	111
0000			1	0	0	0101	0101	01	0	010	000	011	000	000	011
0010			0	0	1	0000	0000	00	0	001	000	000	000	000	001
0110	STRX	1110 0110	1	0	0	0101	0101	01	0	010	000	011	000	000	011
0111	STRX	1110 0110	0	0	1	SR	0000	00	0	001	000	101	000	000	011
0101	STRX	1110 0110	0	0	0	0000	DR	00	0	001	000	011	000	001	000

9.用教学机已实现的基本指令和扩展的几条指令编写程序并运行，测试扩展的几条指令是否正确。

1）测试 ADC 指令。

在命令行提示符状态下输入：

A 2000↙

屏幕将显示：

2000：

从地址 2000H 开始输入下列程序：

2000: MVRD R0,0101 ；给 R0 赋值 0101

2002: MVRD R1,1010 ；给 R1 赋值 1010

2004: ↙

在命令行提示符状态下输入：

A 2006↙

2006: RET

2007: ↙

扩展指令 STC、ADC 不能用 A 命令键入，必须用 E 命令在相应的内存地址键入操作码所有扩展指令都必须用 E 命令键入。

用 E 命令输入 STC、ADC R0,R1 的代码，在命令行提示符状态下输入：

E 2004↙

2004: 6D00

2005：2001

2006: ↙

用 G 命令运行前面刚键入源程序，在命令行提示符状态下输入：

G 2000↙

用 R 命令察看寄存器的内容，在命令行提示符状态下输入

R↙

运行结果应为 R0=1112 R1=1010。

2）测试 JMPR 指令：

在命令行提示符状态下输入：

A 2020↙

屏幕将显示：

2020：

从地址 2020 开始输入下列程序：

```
2020: MVRD R2,000D        ; 给 R2 赋值 000D，000D 为回车键的 ASCII 码值
2022: IN 81               ; 判键盘上是否按了一个键,
2023: SHR  R0             ; 即串行口是否有了输入的字符
2024: SHR R0
2025: JRNC 2022           ; 没有输入则循环测试
2026: IN   80             ; 输入字符到 R0 低位字节
2027：MVRD R1，00FF
2029：AND R0，R1          ; 清零 R0 的高位字节内容
202A: CMP R0,R2           ; 判断输入字符是否为回车
202B: JRZ  2030           ; 若是转向程序结束地址
202C: OUT 80              ; 若否输出键入字符
202D: MVRD R3，2022
202F：↙
```

在命令行提示符状态下输入：

```
A 2030↙
2030: RET
2031: ↙
```

用 E 命令输入 JMPR R3 的代码，在命令行提示符状态下输入：

```
E 202F↙
202F:6003
2030: ↙
```

用 G 命令运行前面刚键入源程序，在命令行提示符状态下输入：

```
G 2020↙
```

光标闪烁等待键盘输入，若输入非回车字符，则在屏幕上回显；若输入回车字符，则程序执行结束。

3）测试 JRS 指令：

在命令行提示符状态下输入：

```
A 2100↙
```

屏幕将显示：

2100：

从地址 2100H 开始输入下列程序：

```
2100：MVRD R1，0000          ；给 R1 赋值 0000
2102：MVRD R2，4040          ；给 R2 赋值 4040
2104：MVRD R3，01FF          ；给 R3 赋值 01FF
2106：ADD R2，R3             ；R2 和 R3 相加
*2107：JRS 210E              ；判第一位，若为 1，向后跳 6 个单元
2108：MVRD R0，0030          ；给 R0 赋字符“0”
210A：OUT 80                 ；输出该字符
```

210B：INC R3　　；R3 加 1
210C：INC R1　　；R1 加 1
210D：JR 2106　　；跳到 2106 循环执行
210E：MVRD R0，0031　　；给 R0 赋字符“1”
2110：OUT 80　　；输出该字符
2111：RET

注：*表示扩展指令 JRS 应用 E 命令键入，在命令行提示符状态下输入：

E 2107↙

2107:6406　　；06 为偏移量，该值是要转向的地址值减去 JRS 下一条指令的地址得出的。

用 G 命令运行前面刚键入源程序，在命令行提示符状态下输入：

G 2100↙

屏幕显示字符 0001。

用 R 命令看寄存器的内容，在命令行提示符状态下输入：

R↙

屏幕回显 15 个寄存器的值，其中 R1 的值表示 R3 加 1 的次数。

可改变 R2、R3 的值观察程序运行结果。以加强对该条指令的理解。

4）测 LDRX、STRX 指令

例 1：测 LDRX 指令.

1〉在命令行提示符状态下输入：

A 2080

屏幕将显示：

2080：

从地址 2080H 开始输入下列程序：

2080：MVRD R2，2000　　；给寄存器 R2 赋值 2000

***2082：LDRX R1，0020[R2]　　；将寄存器 R2 的内容与偏移量相加，相加的和为内存单元 2020，将该单元的内容赋给 R1**

***2084：JMPR R1　　；跳转到寄存器 R1 所示的内存单元**

2085：MVRD R0，0030　　；将字符‘0’的 ASCII 码值赋给 R0

2087：OUT 80　　；输出该字符

2088：RET

2089：↙

注：扩展指令 LDRX、JMPR 必须用 E 命令键入，形式为：E 2082

2082　原值：E512 （空格）　原值：0020（空格）原值：6001↙

2〉在命令行提示符状态下输入：

E 2020

屏幕将显示：

2020　　内存单元原值：-

在光标处输入 2100

3〉在命令行提示符状态下输入：

A 2100

屏幕将显示：

2100：

从地址 2100H 开始输入下列程序：

2100：MVRD R0，0036　　；将字符‘6’的 ASCII 码值赋给 R0

2102：OUT 80　　；输出该字符

2103：RET

2104：↙

4〉在命令行提示符状态下输入：

G 2080↙

屏幕回显数字 6。

例 2：测 STRX 指令.

1〉在命令行提示符状态下输入：

A 2000

屏幕将显示：

2000：

从地址 2000H 开始输入下列程序：

2000:MVRD R1,6666

2002:MVRD R2,2000

*2004:STRX R1，0080[R2]

2006:RET

2007: ↙

扩展指令 STRX 是按如下格式输入的:

在命令行提示符状态下输入：

E 2004↙

2004　内存单元原值：E612（空格） 内存单元原值：0060↙

2〉在命令行提示符状态下输入：

G 2000↙

执行输入的程序。

3〉在命令行提示符状态下输入：

D 2060↙

可以观察到 2060 内存单元的值为 6666，表明寄存器 R1 中的内容已放入该内存单元,内存单元的值是由寄存器 R2 中的内容和偏移量 0060 相加得到的。

2.4 存储器部件教学实验

实验目的

通过看懂教学计算机中已经使用的几个存储器芯片的逻辑连接关系和用于完成存储器容量扩展的几个存储器芯片的布线安排，在教学计算机上设计、实现并调试出存储器容量扩展的实验内容。其最终要达到的目的是：

1. 深入理解计算机内存储器的功能、组成知识；
2. 深入地学懂静态存储器芯片的读写原理和用他们组成教学计算机存储器系统的方法（即字、位扩展技术），控制其运行的方式；

思考并对比静态和动态存储器芯片在特性和使用场合等方面的同异之处。

实验说明

教学计算机存储器系统由 ROM 和 RAM 两个存储区组成，分别由 EPROM 芯片（或 EEPROM 芯片）和 RAM 芯片构成。TEC-XP＋教学计算机中还安排了另外几个存储器器件插座，可以插上相应存储器芯片以完成存储器容量扩展的教学实验，为此必须比较清楚地了解：

1. TEC-XP＋教学机的存储器系统的总体组成及其连接关系；
2. TEC-XP＋教学机的有关存储器芯片、I/O 接口芯片的片选信号控制和读写命令的给出和具体使用办法；
3. RAM 和 EPROM、EEPROM 存储器芯片在读写控制、写入时间等方面的同异之处，并正确建立连线关系和在程序中完成正确的读写过程；
4. 如何在 TEC-XP＋教学机中使用扩展的存储器空间并检查其运行的正确性。

实验内容

1. 要完成存储器容量扩展的教学实验，需为扩展存储器选择一个地址，并注意读写和 OE 等控制信号的正确状态；
2. 用监控程序的 D、E 命令对存储器进行读写，比较 RAM（6116）、EEPROM（28 系列芯片）EPROM（27 系列芯片）在读写上的异同；
3. 用监控程序的 A 命令编写一段程序，对 RAM（6116）进行读写，用 D 命令查看结果是否正确；
4. 用监控程序的 A 命令编写一段程序，对扩展存储器 EEPROM（28 系列芯片）进行读写，用 D 命令查看结果是否正确；如不正确，分析原因，改写程序，重新运行；

实验要求

1. 实验之前，应认真预先准备，写出实验步骤和具体设计内容，否则实验效率会特别低，一次实验时间根本无法完成实验任务，即使基本做对了，也很难说学懂了些什么重要教学内容；
2. 实验过程中，应认真进行实验操作，既不要因为粗心造成短路等事故而损坏设备，又要仔细思考实验有关内容，提高学习的主动性和创造性，把自己想不太明白的问题通过实验理解清楚，争取最好的实验效果，力求达到教学实验的主要目的；

3．实验之后，应认真思考总结，写出实验报告，包括实验步骤和具体实验结果，遇到的主要问题和分析与解决问题的思路。实验报告中，还应写出自己的学习心得和切身体会，也可以对教学实验提出新的建议等。实验报告要交给教师评阅并给出实验成绩。

实验步骤

1．检查 FPGA 下方的标有“/CE”的四组插针均是左边两个短接；

2．检查 RAM（6116）上方的标有“/WE”的插针应是左边两个短接；

3．RAM（6116）支持即时读写，可直接用 A、E 命令向扩展的存储器输入程序或改变内存单元的值。RAM 中的内容在断电后会消失，重新启动实验机后会发现内存单元的值发生了改变。

1〉用 E 命令改变内存单元的值并用 D 命令观察结果。

〈1〉 在命令行提示符状态下输入：

E 2020↙

屏幕将显示： 2020 内存单元原值:

按如下形式键入：

2020 原值：2222 （空格）原值：3333（空格）原值：4444（空格）原值：5555↙

〈2〉 在命令行提示符状态下输入：

D 2020↙

屏幕将显示从 2020 内存单元开始的值，其中 2020H~2023H 的值为：

2222 3333 4444 5555

〈3〉断电后重新启动教学实验机，用 D 命令观察内存单元 2020~2023 的值。会发现原来置入到这几个内存单元的值已经改变，用户在使用 RAM 时，必须每次断电重启后都要重新输入程序或修改内存单元的值。

2〉 用 A 命令输入一段程序，执行并观察结果。

<1> 在命令行提示符状态下输入：

A 2000↙

屏幕将显示： 2000：

按如下形式键入：

2000： MVRD R0，AAAA

2002： MVRD R1，5555

2004： AND R0，R1

2005： RET

2006： ↙

<2> 在命令行提示符状态下输入：

T 2000 ↙

R0 的值变为 AAAAH，其余寄存器的值不变。

T↙

R1 的值变为 5555H，其余寄存器的值不变。

T↙

R0 的值变为 0000H，其余寄存器的值不变。

〈3〉在命令行提示符状态下输入：

G 2000

运行输入的程序。

〈4〉在命令行提示符状态下输入：

R ↙

屏幕显示：

R0=0000 R1=5555 R2=...

RAM 芯片可直接用 A 命令键入程序，但断电会丢失，要再次调试该程序必须重新输入，对那些较长的程序或经常用到的程序可通过交叉汇编，在上位机生成代码文件，每次加电启动后不用重新输入，只需通过 PCEC16 将代码文件传送给下位机即可。

4. 将扩展的 ROM 芯片（27 或 28 系列或 28 的替代产品 58C65 芯片）插入标有“EXTROMH”和“EXTROML”的插座，要注意芯片插入的方向，带有半圆形缺口的一方朝左插入。如果芯片插入方向不对，会导致芯片烧毁。
5. 将扩展芯片下方的插针按下列方式短接：将标有“/MWR” “ PGM”和“RD”的三个插针左面两个短接，将标有“/MWR”“/OE”“GND”的三个插针左边两个短接；
6. 将扩展芯片上方标有 EXTROMH 和 EXTROML 的“/CS”信号用自锁紧线短接，然后短接到 MEMDC 138 芯片的上方的标有“4000－5fff”地址单元；注意：标有/CS 的圆孔针与标有 MEM /CS 的一排圆孔针中的任意一个都可以用导线相连；连接的地址范围是多少，用户可用的地址空间就是多少。
7. 将标有“DataBus 15－8”和“DataBus 7－0”的数据总线的指示灯下方的插针短接；
8. 将标有“AdressBus 15-8”和“AdressBus 7-0”的地址总线的指示灯下方的插针短接；

下面以 2764A 为例，进行扩展 EPROM 实验。

9. EPROM 是紫外线可擦除的电可改写的只读存储器芯片。在对 EPROM 进行重写前必须先擦除并判空，再通过编程器进行编程。

〈1〉将芯片 0000~001F 的内存单元的值置成 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F 10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F

〈2〉将编程好的该芯片插在扩展芯片的高位，低位不插，按上面的提示插好插针。

〈3〉用 D 命令看内存单元 0000~001F 的值。可以看到内存单元的值为：01FF 02FF 03FF 04FF.....1FFF。

〈4〉用 E 命令向芯片的内存单元置值，再用 D 命令察看，会发现原来的值没有改变；用 A 命令向芯片所在的地址键入程序，用 U 命令反汇编，会发现地址仍然保持原来的值。该实验说明 EPROM 不能直接修改和编程，必须先擦除，再通过编程器编程。

〈5〉将教学机断电后重启，用 D 命令看内存单元 0000~001F 的内容，会发现数值没变，EPROM 的内容在断电后会保持。

下面以 AT28C64B（或其替代产品 58C65 芯片）为例，进行扩展 EEPROM 实验。

10. AT28C64B 的读操作和一般的 RAM 一样，而其写操作，需要一定的时间，大约为 1 毫秒。因此，需要编写一延迟子程序，在对 EEPROM 进行写操作时，调用该子程序，以完成正确的读写。

1）用 E 命令改变内存单元的值并用 D 命令观察结果。

〈1〉 在命令行提示符状态下输入：

E 5000↙

屏幕将显示： 5000 内存单元原值：

按如下形式键入：

5000　原值：2424（按空格）原值：3636（按空格）原值：4848（按空格）原值：5050↙

〈2〉 在命令行提示符状态下输入：

D 5000↙

屏幕将显示 5000H~507FH 内存单元的值，从 5000 开始的连续四个内存单元的值依次为 2424 3636 4848 5050。

〈3〉断电后重新启动，用 D 命令察看内存单元 5000~5003 的值，会发现这几个单元的值没有发生改变，说明 EEPROM 的内容断电后可保存。

2） AT28C64B 存储器不能直接用 A 命令输入程序，单字节的指令可能会写进去，双字节指令的低位会出错（建议试一试），可将编写好的程序用编程器写入片内；也可将程序放到 RAM（6116）中，调用延时子程序，访问 AT28C64B 中的内存地址。

下面给出的程序，在 5000H~500FH 单元中依次写入数据 0000H、0001H、...000FH。

从 2000H 单元开始输入主程序：

```
（2000）MVRD R0，0000
        MVRD R2，0010          ；R2 记录循环次数
        MVRD R3，5000          ；R3 的内容为 16 位内存地址
（2006）STRR [R3]，R0          ；将 R0 寄存器的内容放到 R3 给出的内存单元中
        CALA 2200              ；调用程序地址为 2200 的延时子程序
        INC R0                 ；R0 加 1
        INC R3                 ；R3 加 1
        DEC R2                 ；R2 减 1
        JRNZ 2006              ；R2 不为 0 跳转到 2006H
        RET
```

从 2200H 单元开始输入延时子程序：

```
（2200）PUSH  R3
        MVRD R3，FFFF
（2203）DEC R3
        JRNZ 2203
        POP R3
        RET
```

运行主程序，在命令提示符下输入：G 2000↙。

注意：运行 G 命令的时候，必须要将将标有“/MWR”“/OE”“GND”的三个插针右边两个短接。

程序执行结束后，在命令提示符下输入：D　5000↙；

可看到从 5000H 开始的内存单元的值变为

5000：0000 0001 0002 0003 0004 0005 0006 0007

5008：0008 0009 000A 000B 000C 000D 000E 000F。

思考：1）为何能用 E 命令直接写 AT28C64B 的存储单元，而 A 命令则有时不正确；

2）修改延时子程序，将其延时改短，可将延时子程序中 R3 的内容赋成 00FF 或 0FFF 等，再看运行结果。

注意：实验完成后，最好取下扩展芯片，如果芯片插着就最好把片选的线也连接着。

2.5 I/O 口扩展实验

实验目的

学习串行口的正确设置与使用；

实验说明

1. TEC-XP＋配置了两个串行接口 COM1 和 COM2，其中 COM1 口是系统默认的串行口，加电复位后，监控程序对其进行初始化，并通过该口与 PC 机或终端相连；而 COM2 口，留给用户扩展用。
2. 查阅有关书籍，了解串行通信接口芯片 8251 的工作原理；了解 8251 复位、初始化、数据传输的过程。提醒注意的是，每次对 8251 复位后（即按 1 次“RESET”按键），都需要对其进行初始化，然后再进行正常的数据传输；复位后，只能对其进行 1 次初始化，多次初始化将导致串口工作不正常。
3. 在使用 COM2 口时，需要将两片 8251 芯片之间的插针用短路子短接（出厂时已短接），这样才能为 COM2 正常工作提供所需的控制信号和数据；另外，还需要为其分配数据口地址和控制口地址。本教学机，已将 COM2 口的 C/$\overline{D}$ 与地址总线的最低位 AB0 相连，而其片选信号未连，只引出 1 个插孔，实验时，应将该插孔与标有“I/O /CS”的 7 个插孔中的 1 个相连。

实验内容

1. 为扩展 I/O 口选择一个地址，即将与 COM2 口相连的 8251 的/CS 与标有 I/O /CS 的一排插孔中的一个相连；
2. 将 COM2 口与终端或另一台运行有 PCEC16 的 PC 机的串口相连；
3. 用监控程序的 A 命令，编写一段小程序，先初始化 COM2 口，在向 COM2 口发送一些字符，也可从 COM2 口接收一些字符，或实现两个串口的通信；

实验要求

1. 应了解监控程序的 A 命令只支持基本指令，扩展指令应用 E 命令将指令代码写入到相应的存储单元中；

实验步骤

1. 为扩展 I/O 口选择一个地址：将与 COM2 口相连的 SIO2 8251 左上方的/CS 与标有 I/O /CS 的插孔中地址为 A0~AF 的一个相连；
2. 将教学机 COM1 口与一台 PC 机相连，在 PC 机上启动 PCEC16.EXE；
3. 断开 COM1 与 PC 的串口线，将其连接到另一台 PC 机或同一台 PC 的另一个串口，同样启动 PCEC16.EXE；
4. 用另一根串口线将 COM2 口和第一台 PC 或同一台 PC 的另一个串口相连；
5. 在与 COM1 相连的 PCEC 上输入程序，这是主 PCEC 可以输入输出，和 COM2 连接的是从 PCEC 只作输出；
6. 用 A、E 命令编程进行 COM2 口的操作。（标有*的语句要用 E 命令直接写入指令编码）

   1）程序 1：COM2 口初始化

   在命令行提示符状态下输入：

A 2000↙

从2000H单元开始输入下面的程序

```
2000：MVRD R0，004E      ；给R0赋值004E
2002：OUT A1             ；将R0的值输出到COM2口的8251中的寄存器中
2003：MVRD R0，0037      ；给R0赋值0037
2005：OUT A1             ；将R0的值输出到COM2口的8251中的寄存器中
2006：RET
```

在命令行提示符状态下输入G 2000运行初始化程序，完成对COM2口的初始化。

注意：

每次按“RESET”按键后，在对COM2进行读写操作之前，都应运行该程序。需要注意的是，按一次“RESET”按键后，只能对COM2口进行一次初始化操作。

2）程序2：从COM2口输入数据，然后在与COM1口相连的PC上显示出数据。

在命令行提示符状态下输入：

A 2040↙

从2040H单元开始输入下面的程序

```
2040：IN A1              ；  判键盘上是否按了一个键,
2041：SHR R0             ；  即串行口是否有了输入的字符
2042：SHR R0
2043：JRNC 2040          ；  没有输入则循环测试
2044：IN A0              ；  从COM2口读入字符到R0
2045：OUT 80             ；将该字符从COM1口输出
2046：RET
2047：↙
```

运行该程序，在命令行提示符状态下输入：

G 2040↙

光标闪烁等待输入，从与COM2口相连的PC的键盘输入字符，则在与COM1口相连的PC的屏幕上回显。

3）程序3：从COM1口接收数据，发送到与COM2口相连的PC机上回显。

在命令行提示符状态下输入：

A 2060↙

从2060H单元开始输入下面的程序

```
（2060）IN 81            ；判键盘上是否按了一个键,
2061：SHR R0             ；即串行口是否有了输入的字符
2062：SHR R0
2063：JRNC 2060          ；没有，则循环等待
2064：IN 80              ；接收字符
2065：OUT A0             ；将从键盘输入的字符输出到另一串口。
2066：RET
2067：↙
```

运行该程序，在命令行提示符状态下输入：

G 2060↙

光标闪烁等待用户输入，从键盘输入 6，可在另一 PC 的屏幕上回显出 6。

扩展实验也可按另外一种方式完成，操作步骤如下：

1、 为扩展 I/O 口选择一个地址：将与 COM2 口相连的 8251 的/CS 与标有 I/O /CS 的插孔中地址为 A0~AF 的一个相连；注意，将两片 8251 芯片之间的插针短接（出厂时以按默认方式短接）；

2、 将一台教学机 COM1 口与一台 PC 机相连，在 PC 机上启动 PCEC16.EXE；

3、 将另一台教学机 COM1 口与另一台 PC 机相连，同样启动 PCEC16.EXE；

4、 用一根串口线将第一台的教学机的 COM2 口和另一台教学机的 COM2 口相连；

5、 在两台 PC 机对应的 PCEC 上分别输入一下程序：

从 2000H 单元开始输入下面的程序

```
2000：MVRD R0，004E；给 R0 赋值 004E
2002：OUT A1        ；将 R0 的值输出到 COM2 口的 8251 中的寄存器中
2003：MVRD R0，0037；给 R0 赋值 0037
2005：OUT A1        ；将 R0 的值输出到 COM2 口的 8251 中的寄存器中
2006：IN 81         ；检查本机键盘是否按了一个键,
2007：SHR R0        ；即串行口是否有了输入的字符
2008: SHR R0
2009：JRNC 200D     ；没有，则转去检查扩展接口的键盘有没有输入
200A：IN 80         ；若本机键盘有输入则接收该字符
200B: OUT 80        ；将键盘输入的字符在本机输出
200C：OUT A0        ；将从键盘输入的字符输出经扩展串口送到另一台教学机输出
200D: IN A1         ；检查扩展串口相连的另一台教学机对应的 PC 键盘上是否按键,
200E: SHR R0        ；即串行口是否有了输入的字符
200F: SHR R0
2010: JRNC 2006     ；没有，则转去判本机键盘是否有输入
2011：IN A0         ；若有，则接收
2012: OUT 80        ；在本机输出
2013：JR 2006
2014: RET
```

该程序完成两台教学计算机的第 2 个串行接口扩展操作并完成该串口初始化，启动两台教学机，都运行这个程序，则两个键盘的输入同时显示在两个屏幕上，实现的是双机的双向通讯功能。

每台教学机都只能检查与操作自己的串行口，管不了另外那台教学机。

2.6 中断实验

实验目的

学习和掌握中断产生、响应、处理等技术；

实验说明

1. 要求中断隐指令中执行关中断功能，如果用户中断服务程序允许被中断，必须在中断服务程序中执行 EI 开中断命令。
2. 教学机的中断系统共支持三级中断，由三个无锁按键确定从右到左依次为一、二、三级中断，对应的 P1、P0 的编码分别是 01、10、11，优先级也依次升高。这决定了它们的中断向量（即中断响应后，转去执行的程序地址）为 XXX4、XXX8、XXXC；可以看到，每级中断实际可用的空间只有四个字节，故这个空间一般只存放一条转移指令，而真正的用户中断服务程序则存放在转移指令所指向的地址。
3. 用户需扩展中断隐指令、开中断指令、关中断指令、中断返回指令及其节拍。

实验要求

1. 实验前应了解什么是中断向量；
2. 中断隐指令不对应特定指令代码，因而不能用指令代码来判断是否为新指令，在实际设计中，出了复位外，当节拍 T3=1 时，认为是扩展指令。这样在扩展中断隐指令时，节拍 T3 应为 1。

实验内容

1. 扩展中断隐指令，为中断隐指令分配节拍。中断隐指令用到 12 个节拍，为了和一般指令相区别，应将其节拍 T3 设计为 1。注意：在扩展中断隐指令时要用到 DC1、DC2 的译码信号。
2. 扩展开中断指令 EI、关中断指令 DI、中断返回指令 IRET。
3. 确定中断向量表地址。中断向量的高 12 位由开关确定为（0010 0100 0000）。三级中断对应的中断向量为 2404H、2408H、240CH。当有中断请求且被响应后，将执行存放在该中断的中断向量所指向的内存区的指令。
3. 填写中断向量表。在上述的 2404H、2408H、240CH 地址写入三条 JR 转移指令，JR 指令的 OFFSET 是偏移量，其值是要转向的地址的值减去该条转移指令的下一条指令的地址的值得到的，该值的范围在-128~+127 之间。但在 PCEC16 中输入时，用户不需要计算偏移量，直接输入要转向的绝对地址即可。
4. 编写中断服务程序。中断服务程序可以放在中断向量表之后，中断服务程序可实现在程序正常运行时在计算机屏幕上显示与优先级相对应的不同字符；
5. 写主程序。可编写一死循环程序，要求先开中断；

实验步骤

1. 扩展中断隐指令和开、关中断指令、中断返回指令，为他们分配节拍并给出各节拍对应的控制信号。

   中断隐指令的节拍和控制信号

节拍	功能说明	/MIO	REQ	/WE	A	B	Sci	SSH	I8-6	I5-3	I2-0	SST	DC1	DC2
1010	STR→Q. DI#=0	1	0	0	0000	0000	00	00	000	000	111	000	011	111
1110	SP-1→AR、SP	1	0	0	0000	0000	00	00	011	001	011	000	000	011
1100	PC→MEM、INTN#	0	0	0	0101	0000	00	00	001	000	100	000	0011	101
1111	SP-1→AR、SP	1	0	0	0000	0100	00	00	011	001	011	000	000	011
1101	Q→MEM	0	0	0	0000	0000	00	00	001	000	010	000	001	000
1011	INTVL#、IB→PC	1	0	0	0000	0101	00	00	011	000	101	000	101	000

EI、DI、IRET 指令的节拍和控制信号

节拍	指令	编码	/MIO	REQ	/WE	A	B	Sci	SSH	I8-6	I5-3	I2-0	SST	DC1	DC2
1000			1	0	0	0101	0101	01	00	011	001	001	000	000	111
0000			1	0	0	0101	0101	01	00	010	000	011	000	000	011
0010			0	0	1	0000	0000	00	00	001	000	000	000	000	001
0011	EI	0110 1110	1	0	0	0000	0000	00	00	001	000	000	000	110	000
	DI	0110 1111	1	0	0	0000	0000	00	00	001	000	000	000	111	000
0110	IRET	1110 1111	1	0	0	0100	0100	01	00	010	000	011	000	000	011
0100	IRET	1110 1111	0	0	1	0000	0000	00	00	001	000	000	010	000	000
0111	IRET	1110 1111	1	0	0	0100	0100	01	00	010	000	011	000	000	011
0101	IRET	1110 1111	0	0	1	0000	0101	00	00	011	000	111	000	000	000

2. 教师可以参照我们提供的组合逻辑全指令的 MACH 程序，可直接下载的程序到 MACH 器件中。
3. 任意选择几条指令观察指令执行及转中断执行的节拍和各节拍对应的控制信号.

   置控制开关为 11110（单步、手动置指令、组合逻辑、联机、16 位）

   1>选择基本指令的 A 组指令中的 SHR 指令，观察其节拍流程：

   <1>置拨动开关 SW=00001011 00000000；（表示指令 SHR R0 ）

   <2>按 RESET 按键；节拍指示灯 T4~T0 显示 1000；(本拍在第 1 次复位后才会执行)

   <3>按 START 按键；节拍指示灯 T4~T0 显示 0000；(以上两拍,为公共节拍，在手动置指令方式下无意义)

   <4>按 START 按键；节拍指示灯 T4~T0 显示 0010；(本拍也是公共节拍，将指令编码写入指令寄存器 IRH、IRL)

   <5>按 START 按键；节拍指示灯 T4~T0 显示 0011；(本拍执行 SHR 指令，完成 SHR R0 操作)

   <6>按 START 按键；节拍指示灯 T4~T0 显示 0000；(本拍转中断处理,完成 DI#操作)

   <7>按 START 按键；节拍指示灯 T4~T0 显示 0010；(本拍完成中断处理操作 STR→Q)

   <8>按 START 按键；节拍指示灯 T4~T0 显示 0110；(本拍完成中断处理操作 SP-1→AR、SP)

   <9>按 START 按键；节拍指示灯 T4~T0 显示 0100；(本拍完成中断处理操作 PC→（AR），INTN#)

   <10>按 START 按键；节拍指示灯 T4~T0 显示 0111；(本拍完成中断处理操作 SP-1→AR、SP)

   <11> 按 START 按键；节拍指示灯 T4~T0 显示 0101；(本拍完成中断处理操作 Q→（AR))

   <12> 按 START 按键；节拍指示灯 T4~T0 显示 1000；(本拍完成中断处理操作 INTV#，IB→PC)

可以看到，A 组指令（包括 ADD、SUB、CMP、AND、XOR、SHR、SHL、INC、DEC、TEST、OR、MVRR、JR、JRC、JRNC、JRZ、JRNZ）的执行共 11 拍，包括 3 拍公共节拍，一拍完成指令操作，另有 7 拍转中断处理。

2〉选择基本指令的 B 组指令中的 IN 指令，观察其节拍流程：

〈1〉 置拨动开关 SW=10000010 00000000；（表示指令 IN 00 ）

〈2〉 按 RESET 按键；节拍指示灯 T4~T0 显示 1000；(本拍在第 1 次复位后才会执行)

〈3〉 按 START 按键；节拍指示灯 T4~T0 显示 0000；(以上两拍,为公共节拍，在手动置指令方式下无意义)

〈4〉 按 START 按键；节拍指示灯 T4~T0 显示 0010；(本拍也是公共节拍，将指令编码写入指令寄存器 IRH、IRL)

〈5〉 按 START 按键；节拍指示灯 T4~T0 显示 0110；（本拍执行 IN 指令的第一步，IRL→IB→AR）

〈6〉 按 START 按键；节拍指示灯 T4~T0 显示 0100；（本拍执行 IN 指令的第二步，（PORT）→R0）

〈7〉 以下 7 拍转中断处理，节拍流程与 A 组指令的相同。

可以看到，B 组指令（包括 JMPA、LDRR、IN、STRR、PSHF、PUSH、OUT、POP、MVRD、POP、RET）的执行除 3 拍公共节拍外,指令完成需两步，转中断处理后执行 7 步。

3〉选择基本指令的 D 组指令中的 CALA 指令，观察其节拍流程：

〈1〉 置拨动开关 SW=11001110 00000000；（表示指令 CALA）

〈2〉 按 RESET 按键；节拍指示灯 T4~T0 显示 1000；(本拍在第 1 次复位后才会执行)

〈3〉 按 START 按键；节拍指示灯 T4~T0 显示 0000；(以上两拍,为公共节拍，在手动置指令方式下无意义)

〈4〉 按 START 按键；节拍指示灯 T4~T0 显示 0010；；(本拍也是公共节拍，将指令编码写入指令寄存器 IRH、IRL)

〈5〉 按 START 按键；节拍指示灯 T4~T0 显示 0110（本拍执行 CALA 指令的第一步，PC→AR、PC+1→PC）

〈6〉 按 START 按键；节拍指示灯 T4~T0 显示 0100；（本拍执行 CALA 指令的第二步，（AR）→Q）

〈7〉 按 START 按键；节拍指示灯 T4~T0 显示 0111；（本拍执行 CALA 指令的第三步，SP-1→SP、AR）

〈8〉 按 START 按键；节拍指示灯 T4~T0 显示 0101；（本拍执行 CALA 指令的第四步，PC→（AR），Q→PC）

〈9〉 以下 7 拍转中断处理，节拍流程与 A、B 组指令的相同。

4〉选择扩展指令的 A 组指令中的 SBB 指令，观察其节拍流程：

〈1〉 置拨动开关 SW=00100001 00000000；（表示指令 SBB）

〈2〉 按 RESET 按键；节拍指示灯 T4~T0 显示 1000；(本拍在第 1 次复位后才会执行)

〈3〉 按 START 按键；节拍指示灯 T4~T0 显示 0000；(以上两拍,为公共节拍，在手动置指令方式下无意义

〈4〉 按 START 按键；节拍指示灯 T4~T0 显示 0010；(本拍也是公共节拍，将指令编码写入指令寄存器 IRH、IRL)

〈5〉 按 START 按键；节拍指示灯 T4~T0 显示 0011；（本拍执行 SBB 指令，DR-SR-/C→DR）

〈6〉 以下 7 拍转中断处理，节拍流程与基本指令的 A、B 组指令相同

5〉选择扩展指令的 C 组指令中的 LDRA 指令，观察其节拍流程：

〈1〉 置拨动开关 SW=11100100 00000000；（表示指令 LDRA）

〈2〉 按 RESET 按键；节拍指示灯 T4~T0 显示 1000；(本拍在第 1 次复位后才会执行)

〈3〉 按 START 按键；节拍指示灯 T4~T0 显示 0000；(以上两拍,为公共节拍，在手动置指令方式下无意义

〈4〉 按 START 按键；节拍指示灯 T4~T0 显示 0010；(本拍也是公共节拍，将指令编码写入指令寄存器 IRH、IRL)

〈5〉 按 START 按键；节拍指示灯 T4~T0 显示 0110；（本拍执行 LDRA 指令第一步，完成操作 PC→AR，PC+1→PC）

〈6〉 按 START 按键；节拍指示灯 T4~T0 显示 0111；（本拍执行 LDRA 指令第二步，完成操作（AR）→AR）

〈7〉 按 START 按键；节拍指示灯 T4~T0 显示 0101；（本拍执行 LDRA 指令第三步，完成操作（AR）→AR）

〈8〉 以下 7 拍转中断处理，节拍流程与扩展指令的 A 组指令相同

6〉选择扩展指令的 D 组指令中的 IRET 指令，观察其节拍流程：

〈1〉 置拨动开关 SW=11101111 00000000；（表示指令 IRET）

〈2〉 按 RESET 按键；节拍指示灯 T4~T0 显示 1000；(本拍在第 1 次复位后才会执行)

〈3〉 按 START 按键；节拍指示灯 T4~T0 显示 0000；(以上两拍,为公共节拍，在手动置指令方式下无意义

〈4〉 按 START 按键；节拍指示灯 T4~T0 显示 0010；(本拍也是公共节拍，将指令编码写入指令寄存器 IRH、IRL)

〈5〉 按 START 按键；节拍指示灯 T4~T0 显示 0110；（本拍执行 IRET 指令第一步，完成操作 SP→AR，SP+1→SP）

〈6〉 按 START 按键；节拍指示灯 T4~T0 显示 0100；（本拍执行 IRET 指令第二步，完成操作（AR）→FLAG）

〈7〉 按 START 按键；节拍指示灯 T4~T0 显示 0111；（本拍执行 IRET 指令第三步，完成操作 SP→AR，SP+1→SP）

〈8〉 按 START 按键；节拍指示灯 T4~T0 显示 0101；（本拍执行 IRET 指令第四步，完成操作（AR）→PC）

〈9〉 以下 7 拍转中断处理，节拍流程与扩展指令的 A、D 组指令相同

7〉选择几条指令 INC、JRS、MVRD、LDRX、STRX、CALA、IRET，单步运行，观察其各节拍对应的控制信号

节拍	指令	编码	/MIO	REQ	/WE	A	B	Sci	SSH	I8-6	I5-3	I2-0	SST	DC1	DC2
1000			1	0	0	0101	0101	01	00	011	001	001	000	000	111
0000			1	0	0	0101	0101	01	00	010	000	011	000	000	011
0010			0	0	1	0000	0000	00	00	001	000	000	000	000	001
0011	INC	0000 1001	1	0	0	0000	DR	01	00	011	000	011	001	000	000
	JRS	0110 0100	1	0	0	0101	0101	00	00	0S1	000	101	000	010	000
0110	MVRD	1000 1000	1	0	0	0101	0101	01	00	010	000	011	000	000	011
	CALA	1100 1110	1	0	0	0101	0101	01	00	010	000	011	000	000	011
	LDRX	1110 0101	1	0	0	0101	0101	01	00	010	000	011	000	000	011
	STRX	1110 0110	1	0	0	0101	0101	01	00	010	000	011	000	000	011
	IRET	1110 1111	1	0	0	0100	0100	01	00	010	000	011	000	000	011
0100	MVRD	1000 1000	0	0	1	0000	DR	00	00	011	000	111	000	000	000
	CALA	1100 1110	0	0	1	0000	0000	00	00	000	000	111	000	000	000
	IRET	1110 1111	0	0	1	0000	0000	00	00	001	000	000	010	000	000
0111	CALA	1100 1110	1	0	0	0000	0100	00	00	011	001	011	000	000	011
0111	LDRX	1110 0101	0	0	1	SR	0000	00	00	001	000	101	000	000	011
	STRX	1110 0110	0	0	1	SR	0000	00	00	001	000	101	000	000	011
	IRET	1110 1111	1	0	0	0100	0100	01	00	010	000	011	000	000	011
0101	CALA	1100 1110	0	0	0	0101	0101	00	00	010	000	010	000	001	000
0101	LDRX	1110 0101	0	0	1	0000	DR	00	00	011	000	111	000	000	000
	STRX	1110 0110	0	0	0	0000	DR	00	00	001	000	011	000	001	000
	IRET	1110 1111	0	0	1	0000	0101	00	00	011	000	111	000	000	000
10000			1	0	0	0000	0000	00	00	001	000	000	000	000	111
10010			1	0	0	0000	0000	00	00	000	000	111	000	011	000
10110			1	0	0	0000	0100	00	00	011	001	011	000	000	011
10100			0	0	0	0101	0000	00	00	001	000	100	000	001	101
10111			1	0	0	0000	0100	00	00	011	001	011	000	000	011
10101			0	0	0	0000	0000	00	00	001	000	010	000	001	000
11000			1	0	0	0000	0101	00	00	011	000	111	000	101	000

4. 填写中断向量表。

1）将数据开关的高 12 位设置成：0010 0100 0000，即选择 3 级中断的中断向量为 2404H、2408H、240CH。

2）中断向量一共有 16 位，高 12 位由数据开关的 SWH 7-0 以及 SWL7－4 决定；后四位为 P_1P_000，P_1P_0 由按下的无锁按键（中断源）决定，分别为 01、10、11，所以中断向量的 16 位为 2404、2408、240C。

3）填写中断向量表：

从 2404H 单元开始输入下面的程序

```
（2404）JR 2420          ；跳转到中断服务程序
（2408）JR 2430          ；跳转到中断服务程序
（240C）JR 2440          ；跳转到中断服务程序
```

5. 编写中断服务程序。

该中断服务程序，先开中断，显示字符“BI”和对应的中断优先级“1”、“2”或“3”后，等待从键盘输入一个字符。在键盘输入一个字符后，显示该字符和字符“EI”，然后退出当前中断服务程序，返回中断断点，继续执行。

用 A、E 命令从 2420H 单元开始输入下面的程序（标有*的语句表示要用 E 命令输入）

```
2420：PUSH R0            ；R0 进栈
2421：PUSH R3            ；R3 进栈
2422：MVRD R3，31        ；将字符‘1’的 ASCII 码送寄存器 R3
2424：JR 2450

2430：PUSH R0            ；R0 进栈
2431：PUSH R3            ；R3 进栈
2432：MVRD R3，32        ；将字符‘2’的 ASCII 码送寄存器 R3
2434：JR 2450

2440：PUSH R0            ；R0 进栈
2441：PUSH R3            ；R3 进栈
2442：MVRD R3，33        ；将字符‘3’的 ASCII 码送寄存器 R3
2444：JR 2450

*2450:EI                 ；开中断（指令编码：6E00）
2451:MVRD  R0,0042       ；将字符“B”赋值给 R0，B 即 Begin 的缩写。
2453:CALA  2200          ；调用子程序，完成显示
2455:MVRD  R0,0049       ；将字符“I”赋值给 R0，I 即 Interrupt 的缩写。
2457:CALA  2200          ；调用子程序，完成显示
2459:MVRR  R0,R3         ；将 R3 的内容送 R1
245A:CALA  2200          ；调用子程序，完成显示

245C:IN  81              ；判键盘上是否按了一个键
245D:SHR  R0             ；即串口是否有了输入字符
245E:SHR  R0
```

```
245F:JRNC  245C          ；若没有，等待
2460:IN    80            ；输入字符到 R0

2461:MVRD  R0,0045       ；将字符“E”赋值给 R0，E 即 End 的缩写。
2463:CALA  2200          ；调用子程序，完成显示
2465:MVRD  R0,0049       ；将字符“I”赋值给 R0，I 即 Interrupt 的缩 写
2467:CALA  2200          ；调用子程序，完成显示
2469:MVRR  R0,R3         ；将 R3 的内容送 R1
246A:CALA  2200          ；调用子程序，完成显示
246C:POP   R3            ；R3 出栈
246D:POP   R0            ；R0 出栈
*246E:IRET               ；中断返回
```

7.用 A 命令从 2200H 单元开始输入下面的子程序

```
2200:PUSH R0             ；R0 进栈
2201:IN   81             ；查询接口状态，判字符输出完成否
2202:SHR  R0
2203:JRNC 2201           ；未完，循环等待
2204:POP  R0             ；R0 出栈
2205:OUT 80              ；输出 R0 的值
2206:RET
```

8.编写主程序。

从地址 2000H 开始输入下列程序：

```
*2000:EI
 2001:MVRD R0，0036      ；将字符‘6’的 ASCII 码送寄存器 R0
 2003:CALA 2200          ；输出该字符
 2005:MVRD R0,4000       ；延时子程序
 2007:DEC R0
 2008:JRNZ 2007
 2009:JR 2001            ；跳到 2001 循环执行该程序
 200A:RET
```

9、运行主程序，等待、响应中断。

在命令行提示符状态下输入：

G 2000↙

屏幕将连续显示“6”。在程序执行过程中按下教学机右下方任意一个无锁按键。此时，教学机转向执行本级中断服务程序，在屏幕上显示 BI 以及按下的键对应的中断优先级。在接收键盘一个字符后，显示该字符并退出当前级的中断服务程序，恢复中断现场，接着执行断点处的程序。若在接收字符之前，又有更高一级的中断请求，则教学机转向执行高一级的中断服务程序，执行完后接着执行低级中断，然后退出执行主程序。需要注意的是若当前中断为高级的中断，则不会响应低级中断。

2.7 微程序控制器部件教学实验

2.7.1 技术资料汇总

1．指令汇总表

微程序的指令格式、功能、指令分组与组合逻辑的完全相同。指令汇总表参见组合逻辑的指令汇总表。

2．微程序入口地址映射表

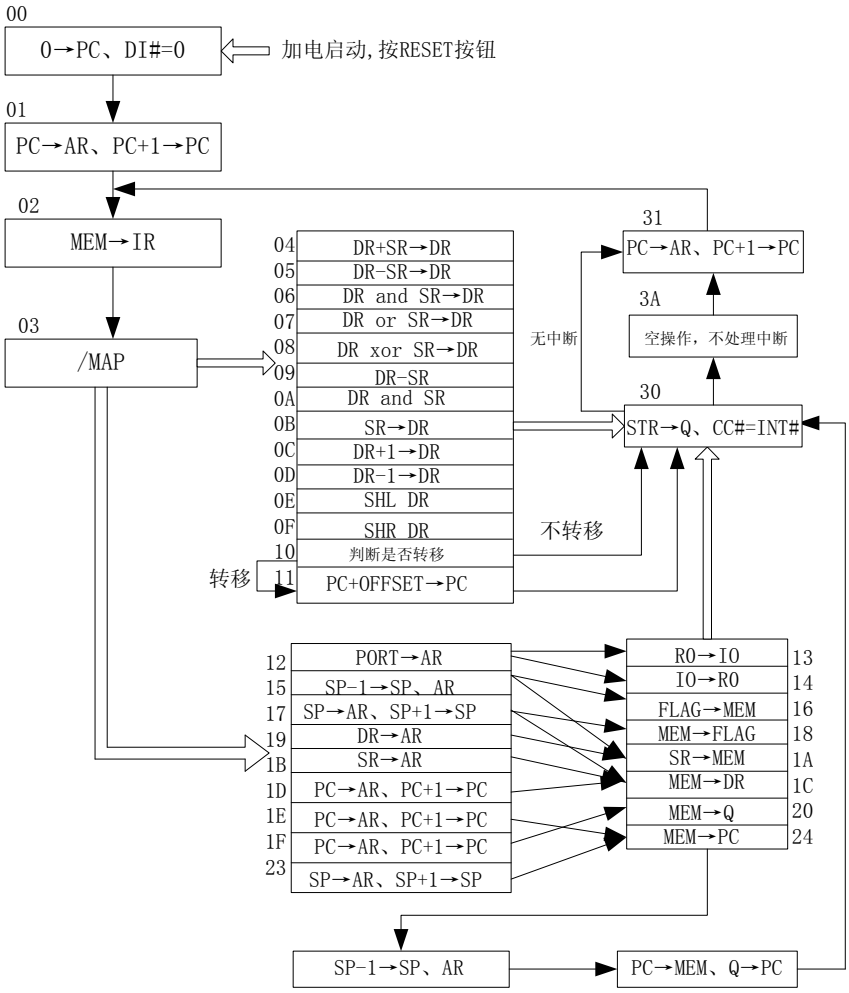
序号	指令	编码	入口地址
1	ADD DR, SR	0000 0000	04
2	SUB DR, SR	0000 0001	05
3	AND DR, SR	0000 0010	06
4	OR DR, SR	0000 0110	07
5	XOR DR, SR	0000 0100	08
6	CMP DR, SR	0000 0011	09
7	TEST DR, SR	0000 0101	0A
8	MVRR DR, SR	0000 0111	0B
9	INC DR	0000 1001	0C
10	DEC DR	0000 1000	0D
11	SHL DR	0000 1010	0E
12	SHR DR	0000 1011	0F
13	JRC OFFSET	0100 0100	10
14	JRNC OFFSET	0100 0101	10
15	JRZ OFFSET	0100 0110	10
16	JRNZ OFFSET	0100 0111	10
17	JR OFFSET	0100 0001	11
18	IN PORT	1000 0010	12
19	OUT PORT	1000 0110	12
20	PSHF	1000 0100	15
21	PUSH DR	1000 0101	15
22	POP DR	1000 0111	17
23	POPF	1000 1100	17
24	STRR [DR], SR	1000 0011	19
25	LORR DR, [SR]	1000 0001	1B
26	MVRD DR, DATA	1000 1000	1D
27	JMPA ADR	1000 0000	1E
28	CALA ADR	1100 1110	1F
29	RET	1000 1111	23
1	ADC DR, SR	0010 0000	50
2	SBB DR, SR	0010 0001	51
3	NOT DR	0010 1101	52
4	ASR DR	0010 1100	53
5	RCL DR	0010 1010	54
6	RCR DR	0010 1011	55
7	CLC	0110 1100	56
8	STC	0110 1101	57
9	EI	0110 1110	58
10	DI	0110 1111	59

11	JMPR SR	0110 0000	5A
12	LDRA DR, [ADR]	1110 0100	5B
13	LDRX DR, OFFSET [SR]	1110 0101	5D
14	STRA [ADR], SR	1110 0111	5F
15	STRX DR, OFFSET[SR]	1110 0110	61
16	CALR SR	1110 0000	64
17	IRET	1110 1111	67
18	JRS OFFSET	0110 0100	69
19	JRNS OFFSET	0110 0101	69

注：该指令入口地址映射表中，前 29 条指令为基本指令，所有基本指令都已编程到微程序控制器中；后 19 条为扩展指令，需用户自己确定完成各步操作所需的控制微码，并将微码扩展到 MAPROM 和七片 MPRAM 中去。在《TEC-XP＋教学计算机系统说明与实验指导》的《微程序控制器》一章中已给出 上的 SCC（GAL20V8）和大板上七片 GAL20V8 的逻辑表达式，其中包括扩展指令，用户在扩展指令时不必改写这八片 GAL20V8 的逻辑表达式。

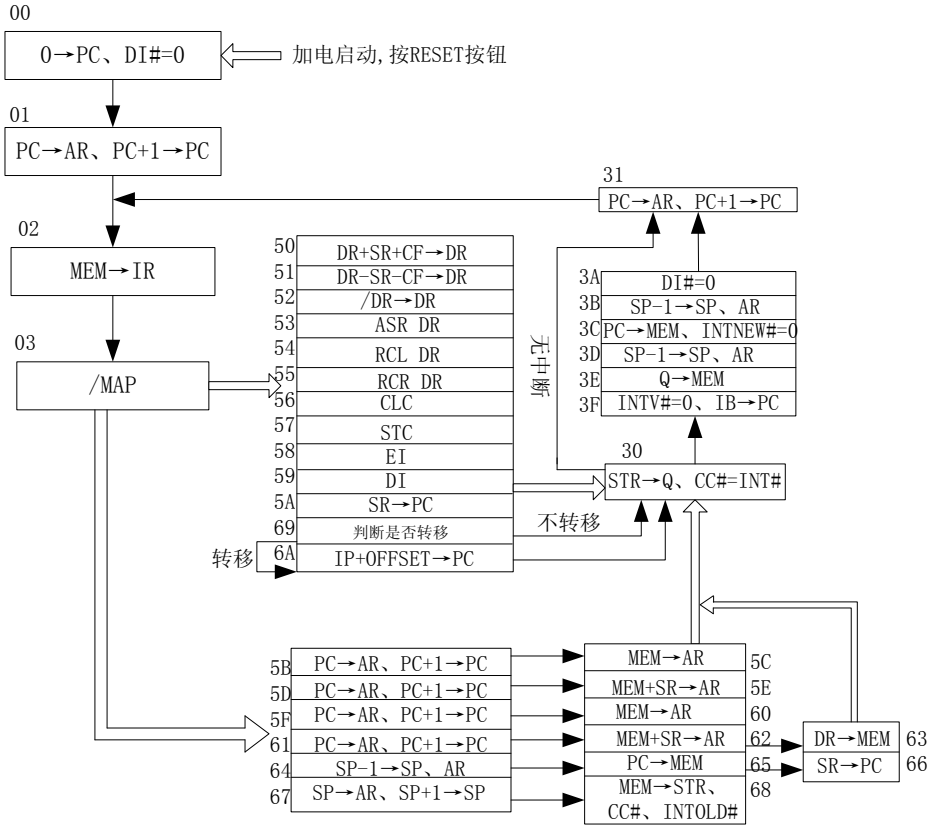
3．指令流程框图

基本指令执行流程框图



注：12 是 IN/OUT 两条指令的入口地址,IN 指令由 12 跳到 14,OUT 指令由 12 跳到 13.
15 是 PUSH/PSHF 两条指令的入口地址,PUSH 指令由 15 跳到 1A,PSHF 指令由 15 跳到 16.
17 是 POP/POPF 两条指令的入口地址,POP 指令由 17 跳到 1C,POPF 指令由 17 跳到 18.
在地址 3A 处放的是一条空操作指令,只起跳转的作用,用户在扩展中断隐指令时,可将该地址的这条指令用中断隐指令代替.

扩展指令执行流程框图


00
0→PC、DI#=0
加电启动,按RESET按钮
01
PC→AR、PC+1→PC
02
MEM→IR
03
/MAP
31
PC→AR、PC+1→PC
50 DR+SR+CF→DR
51 DR-SR-CF→DR
52 /DR→DR
53 ASR DR
54 RCL DR
55 RCR DR
56 CLC
57 STC
58 EI
59 DI
5A SR→PC
69 判断是否转移
转移
6A IP+OFFSET→PC
无中断
3A DI#=0
3B SP-1→SP、AR
3C PC→MEM、INTNEW#=0
3D SP-1→SP、AR
3E Q→MEM
3F INTV#=0、IB→PC
30
STR→Q、CC#=INT#
不转移
5B PC→AR、PC+1→PC
5D PC→AR、PC+1→PC
5F PC→AR、PC+1→PC
61 PC→AR、PC+1→PC
64 SP-1→SP、AR
67 SP→AR、SP+1→SP
MEM→AR 5C
MEM+SR→AR 5E
MEM→AR 60
MEM+SR→AR 62
PC→MEM 65
MEM→STR、CC#、INTOLD# 68
DR→MEM 63
SR→PC 66

4．指令流程表

微程序表

指令	操作功能	微址	下址	CI3~0	SCC30	OMRW	OI2~0	SA、I8~6	SB、I5~3	B 口	A 口	OSST	SSH、SCI	DC2	DC1
ALL	0→PC、DI#=0	00	00	1110	0000	0100	0001	0011	0001	0101	0101	0000	0001	0111	0000
ALL	PC→AR、PC+1→PC	01	00	1110	0000	0100	0011	0010	0000	0101	0101	0000	0001	1011	0000
ALL	MEM→IR	02	00	1110	0000	0001	0000	0001	0000	0000	0000	0000	0000	0001	0000
ALL	/MAP	03	00	0010	0000	0100	0000	0001	0000	0000	0000	0000	0000	0000	0000
ADD	DR+SR→DR	04	30	0011	0000	0100	0001	1011	1000	0000	0000	0001	0000	0000	0000
SUB	DR-SR→DR	05	30	0011	0000	0100	0001	1011	1001	0000	0000	0001	0001	0000	0000
AND	DR∧SR→DR	06	30	0011	0000	0100	0001	1011	1100	0000	0000	0001	0000	0000	0000
OR	DR∨SR→DR	07	30	0011	0000	0100	0001	1011	1011	0000	0000	0001	0000	0000	0000
XOR	DR⊕SR→DR	08	30	0011	0000	0100	0001	1011	1110	0000	0000	0001	0000	0000	0000
CMP	DR-SR	09	30	0011	0000	0100	0001	1001	1001	0000	0000	0001	0001	0000	0000
TEST	DR∧SR	0A	30	0011	0000	0100	0001	1001	1100	0000	0000	0001	0000	0000	0000
MVRR	SR→DR	0B	30	0011	0000	0100	0100	1011	1000	0000	0000	0000	0000	0000	0000
INC	DR+1→DR	0C	30	0011	0000	0100	0011	0011	1000	0000	0000	0001	0001	0000	0000
DEC	DR-1→DR	0D	30	0011	0000	0100	0011	0011	1001	0000	0000	0001	0000	0000	0000
SHL	SHL DR	0E	30	0011	0000	0100	0011	0111	1000	0000	0000	0110	0000	0000	0000
SHR	SHR DR	0F	30	0011	0000	0100	0011	0101	1000	0000	0000	0101	0000	0000	0000
JR CND	JRCnd OFFSET	10	30	0011	0100	0100	0000	0001	0000	0000	0000	0000	0000	0000	0000
JR	Offset+IP→PC	11	30	0011	0000	0100	0101	0011	0000	0101	0101	0000	0000	0000	0010
IN/OUT	PORT→AR	12	14	0011	0110	0100	0111	0001	0000	0000	0000	0000	0000	0011	0010
	R0→IO	13	30	0011	0000	0010	0011	0001	0000	0000	0000	0000	0000	0000	0001
	IO→R0	14	30	0011	0000	0011	0111	0011	0000	0000	0000	0000	0000	0000	0000
PSH/F	SP-1→SP,AR	15	1A	0011	0111	0100	0011	0011	0001	0100	0000	0000	0000	0011	0000
	FLAG→MEM	16	30	0011	0000	0000	0000	0001	0000	0000	0000	0000	0000	0000	0011
POP/F	SP→AR,SP+1→SP	17	1C	0011	0111	0100	0011	0010	0000	0100	0100	0000	0001	0011	0000
	MEM→FLAG	18	30	0011	0000	0001	0000	0001	0000	0000	0000	0010	0000	0000	0000
STRR	DR→AR	19	00	1110	0000	0100	0011	0001	1000	0000	0000	0000	0000	0011	0000
ALL	SR→MEM、CC#=0	1A	30	0011	0000	0000	0100	1001	0000	0000	0000	0000	0000	0000	0001

LORR	SR→AR	1B	00	1110	0000	0100	0100	1001	0000	0000	0000	0000	0000	0011	0000
ALL	MEM→DR、CC#=0	1C	30	0011	0000	0001	0111	0011	1000	0000	0000	0000	0000	0000	0000
MVRD	PC→AR、PC+1→PC、CC#=0	1D	1C	0011	0000	0100	0011	0010	0000	0101	0101	0000	0001	0011	0000
JMPA	PC→AR、PC+1→PC	1E	24	0011	0000	0100	0011	0010	0000	0101	0101	0000	0001	0011	0000
CALA	PC→AR、PC+1→PC	1F	00	1110	0000	0100	0011	0010	0000	0101	0101	0000	0001	0011	0000
	MEM→Q	20	00	1110	0000	0001	0111	0000	0000	0000	0000	0000	0000	0000	0000
	SP-1→SP、→AR	21	00	1110	0000	0100	0011	0011	0001	0100	0000	0000	0000	0011	0000
	PC→MEM、Q→PC、CC#=0	22	30	0011	0000	0000	0010	0010	0000	0101	0101	0000	0000	0000	0001
RET	SP→AR、SP+1→SP	23	00	1110	0000	0100	0011	0010	0000	0100	0100	0000	0001	0011	0000
	MEM→PC、CC#=0	24	30	0011	0000	0001	0111	0011	0000	0101	0000	0000	0000	0000	0000
ALL	STR→Q、CC#=INT#	30	3A	0011	0010	0100	0111	0000	0000	0000	0000	0000	0000	0000	0011
	PC→AR、PC+1→PC、CC#=0	31	02	0011	0000	0100	0011	0010	0000	0101	0101	0000	0001	1011	0000
	用户做中断实验时,自写中断隐指令代替本行微指令.	3A	31	0011	0000	0100	0000	0001	0000	0000	0000	0000	0000	0000	0000
	DI#=0	3A	00	1110	0000	0100	0000	0001	0000	0000	0000	0000	0000	0111	0000
	SP-1→SP、→AR	3B	00	1110	0000	0100	0011	0011	0001	0100	0000	0000	0000	0011	0000
	PC→MEM、INTNEW#=0	3C	00	1110	0000	0000	0100	0001	0000	0000	0101	0000	0000	0101	0001
	SP-1→SP、→AR	3D	00	1110	0000	0100	0011	0011	0001	0100	0000	0000	0000	0011	0000
	Q→MEM	3E	00	1110	0000	0000	0010	0001	0000	0000	0000	0000	0000	0000	0001
	INTV#=0、IB→PC、CC#=0	3F	31	0011	0000	0100	0111	0011	0000	0101	0000	0000	0000	0000	0101
ADC	DR+SR+CF→DR	50	30	0011	0000	0100	0001	1011	1000	0000	0000	0001	0010	0000	0000
SBB	DR-SR-CF→DR	51	30	0011	0000	0100	0001	1011	1001	0000	0000	0001	0010	0000	0000
NOT	/DR→DR	52	30	0011	0000	0100	0011	0011	1111	0000	0000	0001	0000	0000	0000
ASR	ASR DR	53	30	0011	0000	0100	0011	0101	1000	0000	0000	0101	1100	0000	0000
RCL	RCL DR	54	30	0011	0000	0100	0011	0111	1000	0000	0000	0110	0100	0000	0000
RCR	RCR DR	55	30	0011	0000	0100	0011	0101	1000	0000	0000	0101	0100	0000	0000
CLC	CLC	56	30	0011	0000	0100	0000	0001	0000	0000	0000	0011	0000	0000	0000
STC	STC	57	30	0011	0000	0100	0000	0001	0000	0000	0000	0100	0000	0000	0000
EI	EI	58	30	0011	0000	0100	0000	0001	0000	0000	0000	0000	0000	0110	0000
DI	DI	59	30	0011	0000	0100	0000	0001	0000	0000	0000	0000	0000	0111	0000
JMPR	SR→PC、CC#=0	5A	30	0011	0000	0100	0100	1011	0000	0101	0000	0000	0000	0000	0000
LDRA	PC→AR、PC+1→PC	5B	00	1110	0000	0100	0011	0010	0000	0101	0101	0000	0001	0011	0000

	MEM→AR、CC#=0	5C	1C	0011	0000	0001	0111	0001	0000	0000	0000	0000	0000	0011	0000
LDOR	PC→AR、PC+1→PC	5D	00	1110	0000	0100	0011	0010	0000	0101	0101	0000	0001	0011	0000
	MEM+SR→AR、CC#=0	5E	1C	0011	0000	0001	0101	1001	0000	0000	0000	0000	0000	0011	0000
STAR	PC→AR、PC+1→PC	5F	00	1110	0000	0100	0011	0010	0000	0101	0101	0000	0001	0011	0000
	MEM→AR、CC#=0	60	1A	0011	0000	0001	0111	0001	0000	0000	0000	0000	0000	0011	0000
STOR	PC→AR、PC+1→PC	61	00	1110	0000	0100	0011	0010	0000	0101	0101	0000	0001	0011	0000
	MEM+SR→AR、CC#=0	62	00	1110	0000	0001	0101	1001	0000	0000	0000	0000	0000	0011	0000
	DR→MEM	63	30	0011	0000	0000	0011	1001	0000	0000	0000	0000	0000	0000	0001
CALR	SP-1→SP、→AR	64	00	1110	0000	0100	0011	0011	0001	0100	0000	0000	0000	0011	0000
	PC→MEM	65	00	1110	0000	0000	0100	0001	0000	0000	0101	0000	0000	0000	0001
	SR→PC、CC#=0	66	30	0011	0000	0100	0100	1011	0000	0101	0000	0000	0000	0000	0000
IRET	SP→AR、SP+1→SP	67	00	1110	0000	0100	0011	0010	0000	0100	0100	0000	0001	0011	0000
	MEM→STR、CC#、INTOLD#	68	23	0011	0000	0001	0000	0001	0000	0000	0000	0010	0000	0100	0000
JR CND	JRCnd OFFSET	69	30	0011	0101	0100	0000	0001	0000	0000	0000	0000	0000	0000	0000
	Offset+IP→PC	6A	30	0011	0000	0100	0101	0011	0000	0101	0101	0000	0000	0000	0010
Debug 11111111	执行 Debug 前,置 SW=FFFFH,控制台 10001														
	/SWTOIB,D+0->B(R0), Y	70	00	1110	0000	0100	0111	0011	0000	0000	0000	0000	0000	0000	0000
	B(R0)+0->Y,AR,DB, LOOK:Y,AR,DB,IR=FF	71	00	1110	0000	0000	0011	0001	0000	0000	0000	0000	0000	0011	0001
	Cin=1, B(R0)+0+Cin->Y,IR	72	00	1110	0000	0100	0011	0001	0000	0000	0000	0000	0001	0001	0001
	B(R0)+0+Cin->Y,AR,DB, LOOK:Y,AR,DB,IR=00	73	00	1110	0000	0000	0011	0001	0000	0000	0000	0000	0001	0011	0001
	CONTRL=FF	74	00	1110	1111	0111	0111	1111	1111	1111	1111	1111	1111	1111	1111
	CONTRL=00	75	FF	0011	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
		FF	00	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000

2.7.2 微程序控制器实验

在微程序控制器方式下，同样可以做基础汇编语言设计、主存储器扩展、I/O 接口扩展和中断实验。这几项实验的操作步骤与在组合逻辑控制器方式下的实验操作相同，用户可参照前面给出的参考步骤。这里只给出详细的微程序控制器实验的操作步骤。

实验目的

通过看懂教学计算机中已经设计好并正常运行的数条基本指令（例如，ADD、MVRR、OUT、MVRD、JR、RET 等指令）的功能、格式和执行流程，然后自己设计几条指令的功能、格式和执行流程，并在教学计算机上实现、调试正确。其最终要达到的目的是：

1. 深入理解计算机微程序控制器的功能、组成知识；
2. 深入地学习计算机各类典型指令的执行流程；
3. 对指令格式、寻址方式、指令系统、指令分类等建立具体的总体概念；
4. 学习微程序控制器的设计过程和相关技术。

实验说明

控制器设计是学习计算机总体组成和设计的最重要的部分。要在 TEC-XP＋教学计算机完成这项实验，必须比较清楚地懂得：

1. TEC-XP＋教学机的微程序控制器主要由微程序定序器 AM2910、产生当前微地址和下地址的微控存和 MACH 器件组成；
2. TEC-XP＋教学机上已实现的全部基本指令和留给用户实现的 19 条扩展指令的控制信号都是由微控存和 MACH 给出的。
3. 应了解监控程序的 A 命令只支持基本指令，扩展指令应用 E 命令将指令代码写入到相应的存储单元中；不能用 T、P 命令单步调试扩展指令，只能用 G 命令执行扩展指令。
4. 要明白 TEC-XP＋教学机支持的指令格式及指令执行流程分组情况；理解 TEC-XP＋教学机中已经设计好并正常运行的各类指令的功能、格式和执行流程，也包括控制器设计与实现中的具体线路和控制信号的组成。
5. 要明确自己要实现的指令格式、功能、执行流程设计中必须遵从的约束条件。

为了完成自己设计几条指令的的功能、格式和执行流程，并在教学计算机上实现、调试正确的实验内容，具体过程包括：

1> 确定指令格式和功能，包括确定要用的操作码，指令中其它字段的内容分配与使用。指令中字段的使用和分配要受教学机已有硬件的约束，应尽量与已实现指令的格式和分类办法保持一致；

2> 按新指令的功能和格式，设计指令的执行流程。划分指令执行步骤并设计每一步的执行功能，设计微地址和下地址的取值，应参照已实现指令的处理办法来完成；

3> 在指令流程表中填写每一个控制信号的状态值，基本上是个查表填数的过程，应该特别仔细，并有意识地体会这些信号的控制作用；

4> 将设计好的微码，用 ABEL 语言实现出来，可在提供的 MACH 源文件上修改；

5> 写一个包含你设计的指令的程序，通过运行该程序检查执行结果的正确性，来初步判断你的设计是否正确；如果有问题，通过几种办法查出错误并改正，继续调试，直到完全正确。

实验内容

1. 完成控制器部件的教学实验，主要内容是由学生自己设计几条指令的的功能、格式和执行流程，并在教学计算机上实现、调试正确。
2. 首先是看懂 TEC-XP＋教学计算机的功能部件组成和线路逻辑关系，然后分析教学计算机中已经设计好并正常运行的几条典型指令（例如，ADD、MVRR、OUT、MVRD、JRC、CALA、RET 等指令）的功能、格式和执行流程，注意各操作功能所对应的控制信号的作用。
3. 设计几条指令的功能、格式和执行流程，并在教学计算机上实现、调试正确。例如 ADC、JRS、JRNS、LDRA、STAR、CALR 等指令，可以从给出的 19 条扩展指令中任意选择，当然也可以设计与实现其它的指令，包括原来已经实现的基本指令（要变换为另外一个指令操作码）或自己确定的指令。
4. 单条运行指令，查看指令的功能、格式和执行流程。
5. 用监控程序的 A、E（扩展指令必须用 E 命令置入）命令编写一段小程序，观察运行结果。

实验要求

1. 实验之前，应认真准备，写出实验步骤和具体设计内容，否则实验效率会特别低，一次实验时间根本无法完成实验任务，即使基本做对了，也很难说学懂了些什么重要教学内容；
2. 应在实验前掌握所有控制信号的作用，在脱机运算器实验中，已给出了与运算器有关的控制信号的作用，16 位机微程序控制器用到的控制信号的功能表可参见《技术说明与实验指导》的相关内容。

   需要注意的是中断用到了 DC23，在 T4~T0= 0 0010 一拍时 DC23 为 1，其余节拍均为 0；
3. 实验过程中，应认真进行实验操作，既不要因为粗心造成短路等事故而损坏设备，又要仔细思考实验有关内容，提高学习的主动性和创造性，把自己想不太明白的问题通过实验理解清楚，争取最好的实验效果，力求达到教学实验的主要目的；
4. 实验之后，应认真思考总结，写出实验报告，包括实验步骤和具体实验结果，遇到的主要问题和分析与解决问题的思路。大家应该认识到，遇到一些问题是好事情，通过分析与解决这些问题，才提高了自己的工作能力，学习到更多的知识。还未理解清楚，但实验结果正确了就匆忙结束实验，并没有达到教学实验的目的。实验报告中，还应写出自己的学习心得和切身体会，也可以对教学实验提出新的建议等。实验报告要交给教师评阅并给出实验成绩。

实验步骤

1. 接通教学机电源；
2. 将教学机左下方的 6 个拨动开关置为 110100（单步、手动置指令、微程序、联机、16 位、MACH）；
3. 按一下“RESET”按键；
4. 通过 16 位的数据开关 SWH、SWL 置入指令操作码；
5. 在单步方式下，通过指示灯观察**各类基本指令的微码。**

1〉 选择基本指令的 A 组指令中的 **ADD** 指令，观察其节拍流程

〈1〉 置拨动开关 SW=00000000 00000001；（表示指令 ADD R0，R1 ）

〈2〉 按 RESET 按键； 指示灯 Microp 亮（只要选择微程序，该灯在指令执行过程中一直亮），其它灯全灭；

〈3〉 按 START 按键； 指示灯 CI3~0、SCC3~0 显示 1110 0000，微址和下址的指示灯全灭；（本拍完成公共操作 0→PC、DI#=0）

〈4〉 按 START 按键； 指示灯 CI3~0、SCC3~0 显示 1110 0000，微址指示灯显示 0000 0001，下址的指示灯全灭；（本拍完成公共操作 PC→AR、PC+1→PC）

〈5〉 按 START 按键； 指示灯 CI3~0、SCC3~0 显示 1110 0000，微址指示灯显示 0000 0010，下址的指示灯全灭；（本拍完成公共操作 MEM→IR）

〈6〉 以上三步为公共操作，其它指令同；

〈7〉 按 START 按键； 指示灯 CI3~0、SCC3~0 显示 0010 0000，微址指示灯显示 0000 0011，下址的指示灯显示 0000 0100；（本拍完成/MAP 操作功能）

〈8〉 按 START 按键； 指示灯 CI3~0、SCC3~0 显示 0011 0000，微址指示灯显示 0000 0100，下址的指示灯显示 0011 0000（本拍执行 ADD 指令，DR←DR+SR 操作）。

〈9〉 按 START 按键； 指示灯 CI3~0、SCC3~0 显示 0011 0010，微址指示灯显示 0011 0000，下址的指示灯显示 0011 1010；（本拍完成 STR→Q、CC#=INT#公共操作功能）

〈10〉 按 START 按键； 指示灯 CI3~0、SCC3~0 显示 0011 0000，微址指示灯显示 0011 0001，下址的指示灯显示 0000 0010；（本拍完成 PC→AR、PC+1→PC、CC#=0 的公共操作功能）

2〉 选择基本指令的 B 组指令中的 **MVRD** 指令，观察其节拍流程

〈1〉 置拨动开关 SW=10001000 00000000；（表示指令 MVRD ）

〈2〉 按 RESET 按键； 指示灯 Microp 亮（只要选择微程序，该灯在指令执行过程中一直亮），其它灯全灭；

〈3〉 按 START 按键； 指示灯 CI3~0、SCC3~0 显示 1110 0000，微址和下址的指示灯全灭；

〈4〉 按 START 按键； 指示灯 CI3~0、SCC3~0 显示 1110 0000，微址指示灯显示 0000 0001，下址的指示灯全灭；

〈5〉 按 START 按键； 指示灯 CI3~0、SCC3~0 显示 1110 0000，微址指示灯显示 0000 0010，下址的指示灯全灭；

〈6〉 以上三步为公共操作，其它指令同。

〈7〉 按 START 按键； 指示灯 CI3~0、SCC3~0 显示 0010 0000，微址指示灯显示 0000 0011，下址的指示灯显示 0001 1101；

〈8〉 按 START 按键； 指示灯 CI3~0、SCC3~0 显示 0011 0000，微址指示灯显示 0001 1101，下址的指示灯显示 0001 1100；（本拍完成 PC→AR、PC+1→PC、CC#=0 操作）

〈9〉 按 START 按键； 指示灯 CI3~0、SCC3~0 显示 0011 0000，微址指示灯显示 0001 1100，下址的指示灯显示 0011 0000；（本拍完成 MEM→DR、CC#=0 操作）

〈10〉 按 START 按键； 指示灯 CI3~0、SCC3~0 显示 0011 0010，微址指示灯显示 0011 0000，下址的指示灯显示 0011 1010；（本拍完成 STR→Q、CC#=INT#操作）

〈11〉 按 START 按键； 指示灯 CI3~0、SCC3~0 显示 0011 0000，微址指示灯显示 0011 0001，下址的指示灯显示 0000 0010；（本拍完成 PC→AR、PC+1→PC、CC#=0 的公共操作功能）

3〉 选择基本指令的 D 组指令中的 **CALA** 指令，观察其节拍流程

〈1〉 置拨动开关 SW=11001110 00000000；（表示指令 CALA ）

〈2〉 按 RESET 按键； 指示灯 Microp 亮（只要选择微程序，该灯在指令执行过程中一直亮），其它灯全灭；

〈3〉 按 START 按键； 指示灯 CI3~0、SCC3~0 显示 1110 0000，微址和下址的指示灯全灭；

〈4〉 按 START 按键； 指示灯 CI3~0、SCC3~0 显示 1110 0000，微址指示灯显示 0000 0001，下址的指示灯全灭；

〈5〉 按 START 按键； 指示灯 CI3~0、SCC3~0 显示 1110 0000，微址指示灯显示 0000 0010，下址的指示灯全灭；

〈6〉 以上三步为公共操作，其它指令同。

〈7〉 按 START 按键； 指示灯 CI3~0、SCC3~0 显示 0010 0000，微址指示灯显示 0000 0011，下址的指示灯显示 0001 1111；

〈8〉 按 START 按键； 指示灯 CI3~0、SCC3~0 显示 1110 0000，微址指示灯显示 0001 1111，下址的指示灯显示 0000 0000；（本拍完成 PC→AR、PC+1→PC 操作）

〈9〉 按 START 按键； 指示灯 CI3~0、SCC3~0 显示 1110 0000，微址指示灯显示 0010 0000，下址的指示灯显示 0000 0000；（本拍完成 MEM→Q 操作）

〈10〉 按 START 按键； 指示灯 CI3~0、SCC3~0 显示 1110 0000，微址指示灯显示 0010 0001，下址的指示灯显示 0000 0000；（本拍 完成 SP-1→SP、→AR 操作）

〈11〉 按 START 按键； 指示灯 CI3~0、SCC3~0 显示 0011 0000， 微址指示灯显示 0010 0010，下址的指示灯显示 0011 0000；（本拍完成 PC→MEM、Q→PC、CC#=0 操作）

〈12〉 按 START 按键； 指示灯 CI3~0、SCC3~0 显示 0011 0010，微址指示灯显示 0011 0000，下址的指示灯显示 0011 1010；（本拍完成 STR→Q、CC#=INT#操作）

〈13〉 按 START 按键； 指示灯 CI3~0、SCC3~0 显示 0011 0000，微址指示灯显示 0011 0001，下址的指示灯显示 0000 0010；（本拍完成 PC→AR、PC+1→PC、CC#=0 操作）

6. 在连续方式下，用 A 命令健入程序并运行（程序由基本指令组成，可直接用 A 命令键入）。

1〉举例编写汇编程序, 用“A”命令输入,运行并观察结果

例子 1：设计一个小程序, 在屏幕上输出显示字符‘6’。

〈1〉在命令行提示符状态下输入：

A 2000↙ ；

屏幕将显示：

2000：

输入如下形式的程序：

```
2000: MVRD  R0, 0036↙    ; 把字符‘6’的 ASCII 码送入 R0 的低位
2002: OUT  80↙            ; 在屏幕上输出显示字符‘6’
2003: RET↙                ; 每个用户程序都必须用 RET 指令结束
2004: ↙                   ; （按回车键即结束输入过程）
```

〈2〉用“G”命令运行程序

在命令行提示符状态下输入：

G 2000↙

执行上面输入的程序

显示结果为：

6

该例建立了一个从主存 2000H 地址开始的小程序。在这种方式下,所有的数字都约定使用 16 进制数,故数字后不用跟字符 H。每个用户程序的最后一个语句一定为 RET 汇编语句。因为监控程序是选用类似子程序调用方式使实验者的程序投入运行的,用户程序只有用 RET 语句结束,才能保证程序运行结束时能正确返回到监控程序的断点,保证监控程序能继续控制教学机的运行过程。

例 2:设计一个小程序,用次数控制在终端屏幕上输出‘0'到‘9'十个数字符。

<1>在命令行提示符状态下输入：A 2020↙

屏幕将显示：

2020：

从地址 2020H 开始输入下列程序:

```
2020:MVRD R2,000A ; 送入输出字符个数
2022:MVRD R0,0030 ; “0”字符的 ASCII 码送寄存器 R0
2024:OUT 80       ; 输出保存在 R0 低位字节的字符
2025:DEC R2       ; 输出字符个数减 1
2026:JRZ 202E     ; 判 10 个字符输出完否,已完，则转到程序结束处
2027:PUSH R0      ; 未完，保存 R0 的值到堆栈中
2028:IN  81       ; 查询接口状态，判字符串行输出完成否,
2029:SHR R0       ;
202A:JRNC 2028    ; 未完成, 则循环等待
202B:POP R0       ; 已完成, 准备输出下一字符并从堆栈恢复 R0 的值
202C:INC R0       ; 得到下一个要输出的字符
202D:JR  2024     ; 转去输出字符
202E:RET
202F：↙
```

该程序的执行码放在 2020H 起始的连续内存区中。若送入源码的过程中有错, 系统会进行提示, 等待重新输入正确汇编语句。在输入过程中, 在应输入语句的位置直接打回车则结束输入过程。

<2>用“G”命令运行程序

在命令行提示符状态下输入：

G 2020↙

执行结果为：

0123456789

思考题: 当把 IN 01, SHR R0, JNC 2029 三个语句换成 4 个 MOV R0，R0 语句, 该程序执行过程会出现什么现象? 试分析并实际执行一次。

提示：该程序改变这三条语句后，若用 T 命令单条执行，会依次显示 0~9 十个数字。若用 G 命令运行程序，程序执行速度快，端口输出速度慢，这样就会跳跃输出。

在命令行提示符状态下输 G 2020，屏幕显示 09。

类似的，若要求在终端屏幕上输出‘A’到‘Z’共 26 个英文字母，应如何修改例 1 中给出的程序？请验证之。

参考答案：

在命令行提示符状态下输入：

A 2100↙

屏幕将显示：2100：

从地址 2100H 开始输入下列程序：

```
(2100) MVRD R2，001A        ；循环次数为 26
       MVRD R0，0041        ；字符“A”的值
(2104) OUT 80               ；输出保存在 R0 低位字节的字符
       DEC R2               ；输出字符个数减 1
       JRZ  210E            ；判 26 个字符输出完否,已完，则转移到程序结束处
       PUSH R0              ；未完，保存 R0 的值到堆栈中
（2108）IN 81               ；查询接口状态，判字符串行输出完成否
      SHR R0
       JRNC 2108            ；未完成，则循环等待
       POP R0               ；已完成，准备输出下一字符，从堆栈恢复 R0 的值
       INC R0               ；得到下一个要输出的字符
       JR  2104             ；转去输出字符
（210E）RET
```

用 G 命令执行该程序，屏幕上显示“A”~“Z”26 个英文字母.

例子 3:从键盘上连续打入多个属于‘0’到‘9’的数字符并在屏幕上显示,遇非数字符结束输入过程。

<1> 在命令行提示符状态下输入：

A 2040↙

屏幕将显示：

2040：

从地址 2040H 开始输入下列程序:

```
（2040）MVRD   R2,   0030      ；  用于判数字符的下界值
       MVRD   R3,   0039       ；  用于判数字符的上界值
(2044) IN     81               ；  判键盘上是否按了一个键,
       SHR    R0               ；  即串行口是否有了输入的字符
       SHR    R0
       JRNC   2044             ；  没有输入则循环测试
       IN     80               ；  输入字符到 R0
       MVRD R1,00FF
       AND   R0,R1             ；清零 R0 的高位字节内容
```

```
        CMP    R0,    R2          ;  判输入字符≥字符'0'否
    JRNC    2053                ;  为否，则转到程序结束处
    CMP    R3,    R0           ;  判输入字符≤字符'9'否
    JRNC    2053                ;  为否，则转到程序结束处
    OUT    80                  ;  输出刚输入的数字符
    JMPA   2044                ;  转去程序前边 2044 处等待输入下一个字符
(2053)  RET
```

<2>在命令行提示符状态下输入：

G 2040↙

光标闪烁等待键盘输入，若输入 0-9 十个数字符，则在屏幕上回显；若输入非数字符，则屏幕不再显示该字符，出现命令提示符，等待新命令。

思考题，本程序中为什么不必判别串行口输出完成否？设计打入'A'～'Z'和'0'～'9'的程序，遇其它字符结束输入过程。

例子 4：计算 1 到 10 的累加和。

<1>在命令行提示符状态下输入：

A 2060↙

屏幕将显示：

2060：

从地址 2060H 开始输入下列程序：

```
（2060）MVRD    R1,    0000        ;  置累加和的初值为 0
        MVRD    R2,    000A        ;  最大的加数
        MVRD    R3,    0000
(2066)  INC     R3                 ;  得到下一个参加累加的数
        ADD     R1,    R3          ;  累加计算
        CMP     R3,    R2          ;  判是否累加完
        JRNZ    2066               ;  未完，开始下一轮累加
        RET
```

<2>在命令行提示符状态下输入：

G 2060↙

运行过后，可以用 R 命令观察累加器的内容。R1 的内容为累加和。

结果为：R1=0037 R2=000A R3=000A

7．设计几条指令的功能、格式和执行流程，设计每条微指令各字段的具体编码值，包括控制码的各字段、下地址字段、形成下址用到的条件码。

可以从给出的 19 条扩展指令中任意选择，例如 ADC、STC、LDRA、、CALR 等指令，当然也可以设计与实现其它的指令，包括原来已经实现的基本指令（要变换为另外一个指令操作码）或自己确定的指令。

1〉扩展几条指令，确定各步的控制信号。

指令	操作功能	微址	下址	CI3~0	SCC3~0	MRW	I2~0	I8~6	I5~3	B 口	A 口	SST	SSH Sci	DC2	DC1
ADC	DR+SR+CF→DR	50	30	0011	0000	100	001	011	000	0000	0000	001	010	000	000
STC	STC	57	30	0011	0000	100	000	001	000	0000	0000	100	000	000	000
LDRA	PC→AR PC+1→PC	5B	00	1110	0000	100	011	010	000	0101	0101	000	001	011	000
	MEM→AR	5C	1C	0011	0000	001	111	001	000	0000	0000	000	000	011	000
CALR	SP-1→SP、AR	64	00	1110	0000	100	011	011	001	0100	0000	000	000	011	000
	PC→MEM	65	00	1110	0000	000	100	001	000	0000	0101	000	000	000	001
	SR→PC	66	30	0011	0000	100	100	011	000	0101	0000	000	000	000	000

注意：在做扩展指令时，控制信号要由 MACH 产生，教师可以参照提供的微程序的全指令的 MACH 的程序。

2> 将扩展好的控制信号添加到给出的 MACH 程序中，编译生成 JED 的熔丝图文件，写入 MACH 内的寄存器中。教师可以参考提供的微程序全指令的程序,将提供的 MACHM.JED 下载到 MACH 芯片内。(出厂时默认的程序就是微程序全指令的，所以这个步骤可以省略，直接进行下面的步骤)

8. 在单步方式下，通过指示灯观察**各类扩展指令的微码。**

1〉 选择扩展指令的 A 组指令中的 **ADC** 指令，观察其节拍流程

〈1〉 置拨动开关 SW=00100000 00000000；（表示指令 ADC ）

〈2〉 按 RESET 按键； 指示灯 Microp 亮（只要选择微程序，该灯在指令执行过程中一直亮），其它灯全灭；

〈3〉 按 START 按键； 指示灯 CI3~0、SCC3~0 显示 1110 0000，微址和下址的指示灯全灭；

〈4〉 按 START 按键； 指示灯 CI3~0、SCC3~0 显示 1110 0000，微址指示灯显示 0000 0001，下址的指示灯全灭；

〈5〉 按 START 按键； 指示灯 CI3~0、SCC3~0 显示 1110 0000，微址指示灯显示 0000 0010，下址的指示灯全灭；

〈6〉 以上三步为公共操作，其它指令同。

〈7〉 按 START 按键； 指示灯 CI3~0、SCC3~0 显示 0010 0000，微址指示灯显示 0000 0011，下址的指示灯显示 0101 0000；

〈8〉 按 START 按键； 指示灯 CI3~0、SCC3~0 显示 0011 0000，微址指示灯显示 0101 0000，下址的指示灯显示 0011 0000；（本拍完成 DR+SR+CF→DR 操作）

〈9〉 按 START 按键； 指示灯 CI3~0、SCC3~0 显示 0011 0010，微址指示灯显示 0011 0000，下址的指示灯显示 0011 1010；（本拍完成 STR→Q、CC#=INT#操作）

〈10〉按 START 按键； 指示灯 CI3~0、SCC3~0 显示 0011 0000，微址指示灯显示 0011 0001，下址的指示灯显示 0000 0010；（本拍完成 PC→AR、PC+1→PC 操作）

2〉选择扩展指令的 C 组指令中的 **LDRA** 指令，观察其节拍流程

〈1〉 置拨动开关 SW=11100100 00000000；（表示指令 LDRA ）

〈2〉 按 RESET 按键；指示灯 Microp 亮（只要选择微程序，该灯在指令执行过程中一直亮），其它灯全灭；

〈3〉 按 START 按键；指示灯 CI3~0、SCC3~0 显示 1110 0000，微址和下址的指示灯全灭；

〈4〉 按 START 按键；指示灯 CI3~0、SCC3~0 显示 1110 0000，微址指示灯显示 0000 0001，下址的指示灯全灭；

〈5〉 按 START 按键；指示灯 CI3~0、SCC3~0 显示 1110 0000，微址指示灯显示 0000 0010，下址的指示灯全灭；

〈6〉 以上三步为公共操作，其它指令同。

〈7〉 按 START 按键； 指示灯 CI3~0、SCC3~0 显示 0010 0000，微址指示灯显示 0000 0011，下址的指示灯显示 0101 1011；

〈8〉 按 START 按键； 指示灯 CI3~0、SCC3~0 显示 1110 0000，微址指示灯显示 0101 1011，下址的指示灯显示 0000 0000；（本拍完成 PC→AR、PC+1→PC 操作）

〈9〉 按 START 按键； 指示灯 CI3~0、SCC3~0 显示 0011 0000，微址指示灯显示 0101 1100，下址的指示灯显示 0001 1100；（本拍完成 MEM→AR 操作）

〈10〉 按 START 按键； 指示灯 CI3~0、SCC3~0 显示 0011 0000，微址指示灯显示 0001 1100，下址的指示灯显示 0011 0000；（本拍完成 MEM→DR 操作）

〈11〉 按 START 按键； 指示灯 CI3~0、SCC3~0 显示 0011 0010，微址指示灯显示 0011 0000，下址的指示灯显示 0011 1010；（本拍完成 STR→Q 操作）

〈12〉 按 START 按键； 指示灯 CI3~0、SCC3~0 显示 0011 0000，微址指示灯显示 0011 0001，下址的指示灯显示 0000 0010；（本拍完成 PC→AR、PC+1→PC 操作）

3〉 选择扩展指令的 C 组指令中的 **CALR** 指令，观察其节拍流程

〈1〉 置拨动开关 SW=11100000 00000000；（表示指令 CALR ）

〈2〉 按 RESET 按键； 指示灯 Microp 亮（只要选择微程序，该灯在指令执行过程中一直亮），其它灯全灭；

〈3〉 按 START 按键； 指示灯 CI3~0、 SCC3~0 显示 1110 0000，微址和下址的指示灯全灭；

〈4〉 按 START 按键； 指示灯 CI3~0、SCC3~0 显示 1110 0000，微址指示灯显示 0000 0001，下址的指示灯全灭；

〈5〉 按 START 按键； 指示灯 CI3~0、SCC3~0 显示 1110 0000，微址指示灯显示 0000 0010，下址的指示灯全灭；

〈6〉 以上三步为公共操作，其它指令同。

〈7〉 按 START 按键； 指示灯 CI3~0、SCC3~0 显示 0010 0000，微址指示灯显示 0000 0011，下址的指示灯显示 0110 0100；

〈8〉 按 START 按键； 指示灯 CI3~0、SCC3~0 显示 1110 0000，微址指示灯显示 0110 0100，下址的指示灯显示 0000 0000；（本拍完成 SP-1→SP、AR 操作）

〈9〉 按 START 按键； 指示灯 CI3~0、SCC3~0 显示 1110 0000，微址指示灯显示 0110 0101，下址的指示灯显示 0000 0000；（本拍完成 PC→MEM 操作）

〈10〉 按 START 按键； 指示灯 CI3~0、SCC3~0 显示 0011 0000，微址指示灯显示 0110 0110，下址的指示灯显示 0011 0000；（本拍完成 SR→PC 操作）

〈11〉 按 START 按键； 指示灯 CI3~0、SCC3~0 显示 0011 0010，微址指示灯显示 0011 0000，下址的指示灯显示 0011 1010；（本拍完成 STR→Q 操作）

〈12〉 按 START 按键； 指示灯 CI3~0、SCC3~0 显示 0011 0000，微址指示灯显示 0011 0001，下址的指示灯显示 0000 0010；（本拍完成 PC→AR、PC+1→PC 操作）

10.用 A、E 键入程序连续运行（扩展指令用 E 命令键入）。

1〉 测试 ADC 指令。

〈1〉在命令行提示符状态下输入：

A 2000↙

屏幕将显示：

2000：

从地址 2000H 开始输入下列程序：

```
2000: MVRD  R0,0101      ; 给 R0 赋值 0101
2002: MVRD  R1,1010      ; 给 R1 赋值 1010
2004: ↙
```

在命令行提示符状态下输入：

```
A 2006↙
2006: RET
2007: ↙
```

扩展指令 STC、ADC 不能用 A 命令键入，必须用 E 命令在相应的内存地址键入操作码所有扩展指令都必须用 E 命令键入。

用 E 命令输入 STC、ADC R0,R1 的代码，在命令行提示符状态下输入：

```
E 2004↙
2004: 6D00
2005：2001
2006: ↙
```

〈2〉用 G 命令运行前面刚键入源程序，在命令行提示符状态下输入：

G 2000↙

〈3〉用 R 命令察看寄存器的内容，在命令行提示符状态下输入

R↙

运行结果应为 R0=1112 R1=1010。

2>测试 CALR 指令.

设计一个有读写内存和子程序调用指令的程序,功能是读出内存中的字符，将其显示到显示器的屏幕上，转换为小写字母后再写回存储器原存储区域。

〈1〉将被显示的 6 个字符‘A’～‘F’送入到内存 20F0H 开始的存储区域中。

在命令行提示符状态下输入：

E 20F0↙

屏幕将显示：

20F0 内存单元原值：

按下列格式输入：

20F0　内存原值：0041　内存原值：0042　内存原值：0043

内存原值：0044　内存原值：0045　内存原值：0046↙

<2>在命令行提示符状态下输入：

A 2080↙

从地址 2080H 开始输入下列程序：

```
（2080）MVRD  R3, 0006        ；指定被读数据的个数
        MVRD  R2, 20F0        ；指定被读、写数据内存区首地址
（2084）LDRR  R0,[R2]         ；读内存中的一个字符到 R0 寄存器
        MVRD  R8, 2100        ；指定子程序地址为 2100
        *CALR  R8             ；调用子程序，完成显示、转换并写回的功能
        DEC   R3              ；检查输出的字符个数
        JRZ    208C           ；完成输出则结束程序的执行过程
        INC   R2              ；未完成，修改内存地址
        JR    2084            ；转移到程序的 2086 处，循环执行规定的处理
（208C）RET
```

从地址 2100H 开始输入下列程序：

```
（2100）OUT   80              ；输出保存在 R0 寄存器中的字符
        MVRD  R1，0020
        ADD   R0，R1          ；将保存在 R0 中的大写字母转换为小写字母
        STRR [R2],R0          ；写 R0 中的字符到内存，地址同 LOD 所用的地址
（2105）IN   81               ；测试串行接口是否完成输出过程
        SHR   R0
        JRNC  2105            ；未完成输出过程则循环测试
        RET                   ；结束子程序执行过程，返回主程序
```

<3>在命令行提示符状态下输入：

G 2080↙

屏幕显示运行结果为：

ABCDEF

<4>在命令行提示符状态下输入：

D 20F0↙

20F0H~20F5H 内存单元的内容为：

0061　0062　0063　0064　0065　0066

3> 测试指令 LDRA.

编写一程序，将存放在地址单元 2100~2101 的内容通过 LDRA 指令转到寄存器 R0 输出。

<1>　将要输出的字符存放在地址单元 2100。

在命令行提示符状态下输入：

E 2100↙

屏幕将显示：

2100　　内存单元原值：

按下列格式输入：

2100　内存原值：　0036　　内存原值：0038↙

〈2〉用 A、E 命令键入程序。

在命令行提示符状态下输入：

A 2000↙

从地址 2000H 开始输入下列程序：

```
2000：*LDRA R0，[2100]      ；将内存单元 2100 的内容赋给寄存器 R0
2002：OUT 80                ；输出 R0 的内容
2003：IN 81                 ；判是否输出完
2004：SHR R0
2005：JRNC 2003             ；未完，则循环测试
2006：*LDRA R0，[2101]      ；将内存单元 2101 的内容赋给寄存器 R0
2008：OUT 80                ；输出 R0 的内容
2009：RET
```

带*的两条指令为扩展指令，必须用 E 命令键入，形式如下：

E 2000↙

2000　内存单元原值：E400　　内存单元原值：2100↙

E 2006↙

2006　内存单元原值：E400　内存单元原值：2101↙

〈3〉在命令行提示符状态下输入：

G 2000↙

屏幕将回显字符‘6’‘8’

注：做微程序中断实验，用户要先将中断中断隐指令、开中断指令、关中断指令、中断返回指令扩展到 MAPROM、CMH、CML 和 MACH 中去,扩展的步骤与前面介绍的扩展指令过程相同。指令的功能和控制信号的编码参见前面给出的方案。中断服务程序和主程序参见组合逻辑控制器中断实验部分。

2．8　BASIC 语言程序设计

实验目的：

1．学习和了解用于 BASIC 解释执行程序中的子程序；

2．学习和理解计算机软件系统的层次结构；

3．了解高级语言和汇编语言在处理能力和使用的方便程度等方面的区别。

实验内容：

1．了解并练习使用 BASIC 的基本命令；

2．了解解释程序已经实现的语句和对表达式的支持；

3．使用这些知识练习编写简单的典型 BASIC 程序。

实验要求

在使用该教学机之前，应先熟悉教学机的各个组成部分，及其使用方法。

实验步骤

一．实验具体操作步骤：

1．将 TEC-XP＋实验系统左下方的六个黑色开关置为 000100（连续、内存读指令、微程序、联机、16 位、MACH）；

2．打开电源，船形开关和 5V 电源指示灯亮。

3．在 PC 机上运行 PCEC16.EXE 文件，根据连接的 PC 机的串口设置所用 PC 机的串口为“1”或“2”，其它的设置一般不用改动,直接回车即可。(具体步骤附后)

4．按一下“RESET”按键,再按一下“START”按键，主机上显示：

```
TEC-2000 CRT MONITOR
 Version 1.0 April 2001
 Computer Architectur Lab.,  Tsinghua University
         Programmed by He Jia
>
```

5．在命令行提示符下输入 G 0A30 回车；

6．在屏幕出现的：命令行提示符下分别输入下面给出的 6 个小程序的例子；

包括实现整数排序、河内塔问题、求素数的问题、8 皇后问题、验证歌德巴赫猜想、计算三角（正弦）函数等功能的程序，请注意，这些程序最终是通过教学计算机的指令系统执行的。给出这几个程序的目的，是让大家初步学习使用最简单的高级语言完成程序设计的过程和方法，体会高级语言和汇编语言在的语句格式和功能等方面的区别，体会使用高级语言设计程序的优越性。

例 1．这是一个完成整数排序功能的程序，要求首先输入 5 个参加排序的整数数值，接下来完成对这 5 个整数的排序操作，并输出最终的排序结果。

<1>在命令行提示符：下输入下面程序：

```
10  for i=1 to 5
20  input a(i)
30  next i
40  for i=1 to 4
50  for j=i+1 to 5
60  if a(i)>a(j)  then  b=a(i) : a(i)=a(j) : a(j)=b
70  next j
80  next i
90  for i=1 to 5
100  print a(i)
110  next i
120  end
```

<2> 输入 list 命令察看输入的内容是否正确，如果不正确可以直接修改错误的语句，只要在提示符后重新输入标号及其对应的正确内容即可；

<3>输入 run 命令运行程序，结果显示到屏幕上；
<4>输入 system 命令退出 basic 解释程序，返回到教学机监控。

例 2． 这是一个求素数的程序，即在指定的数据(100)范围内，找出除了能被 1 和这个数本身整除之外，不会再被另外的数整除的全部正整数，并将结果显示在计算机屏幕上。
<1>在命令行提示符：下输入下面程序：

```
10  dim a(100)
20  for i=2 to 100
30  j=i
40  j=j+1
50  if  i*j<100 then a(i*j)=1 ： goto 40
60  next i
70  for i=2 to 99
80  if a(i)=0 then print i,
90  next i
100  end
```

<2> 输入 list 命令察看输入的内容是否正确，如果不正确可以直接修改错误的语句，只要在提示符后重新输入标号及其对应的正确内容即可；
<3>输入 run 命令运行程序，结果显示到屏幕上；
<4>输入 system 命令退出 basic 解释程序，返回到教学机监控。

例 3． 这是一个解决河内塔问题的程序，要求把从大到小自底向上叠起来的几个盘子，从当前位置移动到另外一个位置，条件是必须保证在任何时刻不得出现大盘子压在小盘子上面的情形，在移动的过程中，还会用到另外一个缓冲位置，以便临时存放中间结果。

<1>在命令行提示符：下输入下面程序：

```
10  dim src(10),  dst(10),  tmp(10)
20  input n
30  i=n ： src(n)=1 ： dst(n)=2 ： tmp(n)=3
40  i=i-1 ： src(i)=src(i+1 ) ： dst(i)=tmp(i+1) ： tmp(i)=dst(i+1)
50  if i>0 then 40
60  i=i+1 ： if i>n then end
70  if src(i-1)<>src(i) then 60
80  print src(i);  “--- >”;  dst(i),
90  i=i-1 ： src(i)=tmp(i+1) ： dst(i)=dst(i+1) ： tmp(i)=src(i+1)
100  if i>0 then 40
110  goto 60
```

<2> 输入 list 命令察看输入的内容是否正确，如果不正确可以直接修改错误的语句，只要在提示符后重新输入标号及其对应的正确内容即可；
<3>输入 run 命令运行程序，结果显示到屏幕上；
<4>输入 system 命令退出 basic 解释程序，返回到教学机监控。

例 4． 这是一个解决 8 皇后问题的程序，是在 8 行*8 列的棋盘上，以相互不能“吃子”的方式放进 8 个皇后棋子，即在任何一个横排上、任何一个竖列上、任何一个对角线的方向上，都不得同时出现两个皇后棋子，把全部可行结果排列出来，并显示在计算机屏幕上。

<1>在命令行提示符：下输入下面程序：

```
10  dim colstate(7), fdiagstate(14), bdiagstate(14), queenpos(7)
20   i=0 : count=0
30  queenpos(i)=0
40  if colstate(queenpos(i))+fdiagstate(i-queenpos(i)+7)+bdiagstate(i+queenpos(i)) >0 then 170
50  colstate(queenpos(i))=1 : fdiagstate(i-queenps(i)+7)=1 : bdiagstate(i+queenpos(i))=1
60  if i<7 then i=i+1 : goto 30
70   count=count+1
80  print : print ”Result:” ; count; ”:”
90   j=0
100  k=0
110   if k=queenpos(j) then print “0” ; : goto 130
120 print “.” ;
130 k=k+1 : if k<8 then 110
140 print
150 j=j+1 : if j<8 then 100
160 colstate(queenpos(i))=0 : fdiagstate(i-queenps(i)+7)=0 : bdiagstate(i+queenpos(i))=0
170 queenps(i)= queenps(i)+1 : if queenpos(i)<8 then 40
180 i=i-1 : if i>=0 then 160
190 print : print “Total results:” ; cout
200 end
```

<2> 输入 list 命令察看输入的内容是否正确，如果不正确可以直接修改错误的语句，只要在提示符后重新输入标号及其对应的正确内容即可；

<3>输入 run 命令运行程序，结果显示到屏幕上；

<4>输入 system 命令退出 basic 解释程序，返回到教学机监控。

例 5． 这是一个在数值 100 范围内验证歌德巴赫猜想的程序，即任何一个大于 2 的偶数都等于另外两个素数之和，把验证的结果显示在计算机屏幕上。

<1>在命令行提示符：下输入下面程序：

```
10  for i=4 to 100 step 2
20  for j=i to i-2
30  n=j
40  gosub 200
50  if p=0 then 100
60  n=i-j
70  gosub 200
80  if p=0 then 100
```

```
90  print I ; “ =” ; “ +” ; n ,  :  goto 100
100  next j
110  next i
120  end
200  p=0
210  if n/2 <2 then 250
220  for k=2 to n/2
230  if n mod k =0 then 260
240  next k
250  p=1
260  return
```

<2> 输入 list 命令察看输入的内容是否正确，如果不正确可以直接修改错误的语句，只要在提示符后重新输入标号及其对应的正确内容即可；

<3>输入 run 命令运行程序，结果显示到屏幕上；

<4>输入 system 命令退出 basic 解释程序，返回到教学机监控。

例 6． 这是计算正弦三角函数的程序，将 0~360 度范围内的正弦曲线显示在计算机屏幕上。

<1>在命令行提示符：下输入下面程序：

```
10  pi=3.14159
20  for i=0 to 20
30  angle=pi*i/10
40  for  j=1 to 40+25*sin(angle)
50  pring “ ” ;
60  next j
70  print “*”
80  next i
90  end
```

<2> 输入 list 命令察看输入的内容是否正确，如果不正确可以直接修改错误的语句，只要在提示符后重新输入标号及其对应的正确内容即可；

<3>输入 run 命令运行程序，结果显示到屏幕上；

<4>输入 system 命令退出 basic 解释程序，返回到教学机监控。

注意：运行 BASIC 程序必须是在全指令集下，也就是 MACH 芯片内的下载的是微程序全指令或者组合逻辑全指令的程序。

2．9 FPGA 芯片实现非流水线的 CPU 系统

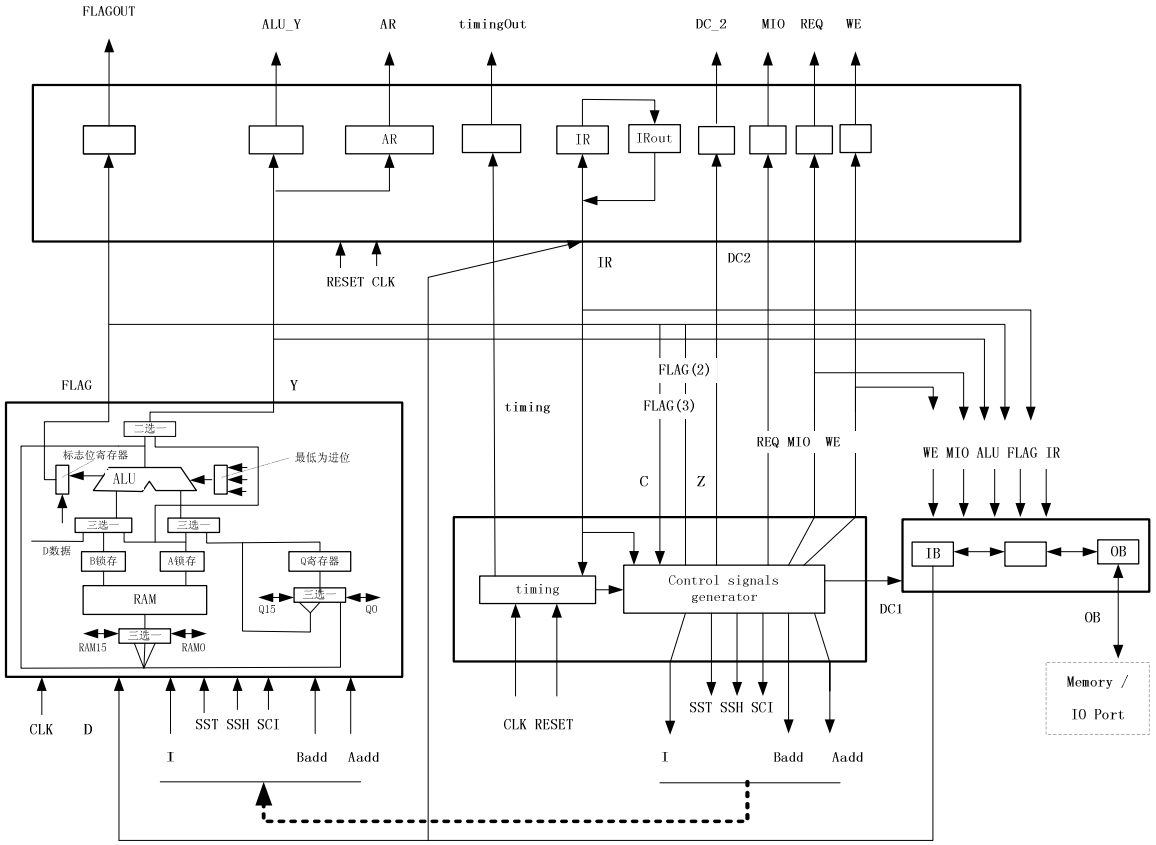
进入到 21 世纪，随着半导体集成电路的迅猛发展，人们对专用集成电路（ASIC）设计的需求与期望值越来越高，希望能够在单个电路芯片上实现一个系统的全部功能。为此，我们利用 VHDL 语言进行描述，通过 FPGA 门阵列器件硬件实现了一个 16 位字长的 CPU 系统。该系统与存储器和输入输出接口线路相连接，共同组成了一台用于硬件课程教学的完整计算机系统。

新设计与实现的 CPU 在指令系统、使用的软件资源等方面与小规模器件构成的左边的教学计算机系统完全兼容；在硬件构成、实现技术等方面也似。这样可以使软件资源得到充分地应用，减轻研制软件系统的负担，又能更好地在两个系统之间得到尽可能高的可比较性，降低授课难度，提高学生的学习效率。所以新的设计与实现的 CPU 系统的外特性是严格限定了的，它与小规模器件构成 cpu 的教学计算机是严格意义上的同一体系结构的计算机，差别仅表现在计算机的具体实现有所不同，包括选用的器件的类型和集成度不同，所用的设计手段、设计过程有所不同，体现出来的设计与实现技术也不尽相同。

该系统选择了 xinlinx 公司的 SPARTAN—II系列的芯片（型号是 XC2S200），20 万门容量，其内部有 2352 个 CLB，14 个 4Kb 的 RAM 块，208 脚的 PQFP 封装形式，支持在系统编程（in-system programmable），实现了 CPU 的全部功能。在完成这项任务时，已经考虑到如何照顾到现有教材和试验指导书内容的稳定性。首先，需要保证新设计的教学机的指令系统，与过去已经使用的 TEC-2000 教学计算机的指令系统有良好的兼容性。其次，在构思新型教学计算机的逻辑结构的过程中，要向原 TEC-2000 教学计算机的实际组成适当地靠拢，尽量地在二者之间有一个平滑的过渡。这对我们们的设计，加进了某些限制条件，但是对减轻任课教师的教学负担、保证教学质量是至关重要的。

当把选用 VHDL 语言来描述的 CPU 的源码文件，经过专用工具软件的编译和综合后，下载到这样一个 FPGA 芯片之中，也就得到了能够正常运行的 CPU 系统。

芯片内部的功能结构图如下：



一. 实验目的

1. 进一步熟悉教学计算机的指令格式、指令编码、寻址方式和指令功能等内容；
2. 进一步熟悉教学计算机的总体组成和各个部件的功能，理解控制器部件在计算机整机中的关键作用；

3. 进一步理解和熟悉指令执行步骤的划分方案；
4. 进一步熟悉教学计算机的硬连线控制器各个控制命令（组）的控制功能，学习用 VHDL 语言描述节拍发生器和控制信号产生部件的功能。
5. 进一步理解与熟悉在 TH-union 教学计算机控制器中处理原有指令和扩展指令的方案，提高对控制器功能能描述的理解程度。

二．实验内容

控制器实验可以在两个层次上进行：

第一个层次属于观察、验证性的实验，即通过多种方式，察看教学计算机指令的执行步骤、运行结果、各组控制信号在每一个执行步骤中的状态、指令之间的衔接等有关内容。这个层次的实验，重点在于学懂教学计算机中已有的设计结果，把实现基本指令的 VHDL 语言程序中的语句描述与教学计的运行结果对应清楚。

第二个层次是学生进行自己的设计与实现新的扩展指令的实验，即在教学机系统已有指令的基础上，由学生自己添加若干条（例如 3～5 条）新的指令进去，包括定义指令格式、功能，划分指令执行步骤和确定每一步的功能，确定每一执行步骤使用的全部控制信号的状态值，使用 VHDL 语句把新的设计结果描述正确并添加到已有的源程序代码中去，编译、下载并调试正确，写一个包含已有指令和刚刚实现的指令的小程序，检查运行结果的正确性，若发现错误则找出原因并设法改正，直到全部正确为止。

学生扩展实现哪几条指令，可以由教师指定，也可以由学生根据自己学习情况自选另外一、两条。这些指令最好从教材中给出的扩展指令组中挑选，例如 2 条 A 组指令和 2 条 C 组指令。在设计指令的操作码编码、指令执行步骤、使用的控制信号等方面，尽可能地参照已有的基本执行的实现办法进行类似的处理，有利于降低实验难度。

三．实验步骤

1. 按前述的步骤准备好实验机，连接好串口线和电源线，打开 PCEC16.EXE 的仿真界面；
2. 将六个功能开关置为 00X101（连续、内存读指令、连机、16 位、FPGA）；
3. 确认标有“DataBus 15－8”和“DataBus 7－0”的数据总线的指示灯下方的插针断开；
4. 确认标有“AdressBus 15-8”和“AdressBus 7-0”的地址总线的指示灯下方的插针断开；
5. 将提供的带彩线的 FPGA 的下载线并口一端和计算机的并口连接，彩线一端按红色在左边的位置和大板上电源模块下方的一排插针插接好；
6. 打开实验机的电源；
7. 在 PC 机上打开 ISE 的软件（软件的具体编译下载使用参见光盘附录）
8. 打开软件的下载界面，选择 SLAVE SERIAL 方式，添加器件 CPU.BIT，进行下载；
9. 下载完成关闭下载界面，启动 PCEC 界面，注意实验机不要断电（FPGA 断电丢失内容）；
10. 按一下“RESET”按键,再按一下“START”按键，主机上显示：

```
TEC-2000 CRT MONITOR
 Version 1.0 April 2001
 Computer Architectur Lab.,  Tsinghua University
          Programmed by He Jia
>
```

11. 在 FPGA 构成的 CPU 的控制下将汇编语言程序设计的内容重新作一遍。