返回 主页

常见问题

1. 一条指令的执行过程中要做哪些事情呢?

答：一条指令的执行过程包括：取指令、指令译码、（计算操作数地址）、取操作数、运算、送结果。其中取指令和指令译码是每条指令都必须进行的操作。有些指令需要到内存单元取操作数，因此，需要在取数之前计算操作数的内存单元地址。取操作数和送结果这两个步骤，对于不同的指令，其取和送的地方可能不同，有些指令要求在寄存器取/送数，有些是在内存单元取/送数，还有些是对I/O端口取/送数。因此，一条指令的执行阶段（不包括取指令阶段），可能只有CPU参与，也可能要通过总线去访问主存，也可能要通过总线去访问I/O端口。

2. 指令周期、机器周期之间的关系是什么?(问题中的“CPU周期”去掉)

答：一条指令从读取到执行完成所花的全部时间被称为指令周期。一个指令周期中要完成多个步骤，包括取指令、指令译码、（计算操作数地址）、取操作数、运算、送结果。这些步骤中，最复杂的操作是访问存储器取指令或读/写数据，以及访问I/O读/写数据。它们都涉及到总线操作，通过系统总线来和CPU之外的部件进行信息交换。通常把通过一次总线事务访问一次主存或I/O的时间称为机器周期。所以一个指令周期包含了多个机器周期。不同机器的指令周期所包含的机器周期数不同。典型的机器周期有：取指令、主存读（间址周期是一种主存读机器周期）、主存写、I/O读、I/O写、中断响应等。

3. CPU总是在执行指令吗？会不会停下来什么都不做?

答：CPU的功能就是不断地周而复始地执行指令，而每条指令又都有不同的步骤，每个步骤在一定的时间内完成。因此，CPU总是在不停地执行指令。有时我们会说，CPU停止或CPU正在等待，什么事情也不做。事实上，CPU还是在执行指令的，只不过可能处于以下几种类似的情况：(1) 在执行指令过程中，正在等待主存或I/O完成读/写；(2) 正在执行一连串的空指令（NOP）；（3）可能正在执行一个循环（循环内只是不断地取状态、判断、不满足时继续循环），直到满足某个条件（如：查询外设有没有完成任务）；等等。因而，CPU不可能不在执行指令。

4. CPU除了执行指令外，还做什么事情?

答：CPU的工作过程就是周而复始地执行指令，计算机各部分所进行的工作都是由CPU根据指令的要求来启动的。为了使CPU和外部设备能够很好地协调工作，尽量使CPU不等待、甚至不参与外部设备的输入和输出过程，采用了程序中断方式和DMA方式。这两种方式下，外部设备需要向CPU提出中断请求或DMA请求，因此，在执行指令过程中，CPU还要按时通过采样相应的引脚来查询有没有中断请求或DMA请求。一般，在一个机器周期结束时，查询是否有DMA请求，如果有的话，CPU脱离总线，由DMA控制器控制使用总线。在一个指令周期结束时，查询是否有中断请求，如果有的话，则进入中断响应机器周期，相当于执行了一条中断响应隐指令。在中断响应过程中，得到中断服务程序的入口地址，并送程序计数器PC中，下个指令周期开始时，取出中断服务程序的第一条指令执行。

5. CPU中的所有寄存器，用户都能访问吗?

答：CPU中的寄存器分为用户可访问寄存器和用户不可见寄存器。一般把用户可访问寄存器称为通用寄存器（GPR）。这些寄存器都有一个编号，在指令中用编号标识寄存器。所以执行指令时，指令中的寄存器编号要送到一个地址译码器进行译码，然后才能选中某个寄存器进行读写。通用寄存器可以用来存放操作数或运算结果，或作为地址指针、变址寄存器、基址寄存器等。

CPU中有一些寄存器是用户不可见的，没有编号、不能通过程序直接访问。如：程序计数器PC、指令寄存器IR、程序状态字寄存器PSWR、存储器地址寄存器MAR、存储器数据寄存器MDR等。

6. CPU执行指令的过程中，其他部件在做什么?

答：计算机的工作过程就是连续执行指令的过程，整个计算机各个部分的动作都是由CPU中的控制部件CU通过对指令译码送出的控制信号来控制的。其他部件不知道自己该做什么，该完成什么动作，只有CPU通过对指令译码才知道。如果指令中包含有对存储器或I/O模块的访问，则必须由CPU通过总线，把要访问的地址和操作命令（读还是写）等信息送到存储器或I/O接口（I/O模块）来启动相应的读或写操作。例如，每次指令执行前，都要通过向总线发出主存地址、主存读命令等来控制存储器取指令；若当前执行的是寄存器定点加法指令，则CU控制定点运算器进行动作；若是I/O指令，则CU会通过总线发出I/O端口地址、I/O读或写命令等来控制对某个I/O接口中的寄存器进行读写操作。所以说，CPU在执行指令时，其他部件也在执行同样的指令，只不过一条指令中包含了不同的动作，在不同的地方执行而已。

7. 怎样保证CPU能按程序规定的顺序执行指令呢?

答：计算机的工作过程就是连续执行指令的过程，指令在主存中连续存放。一般情况下，指令被顺序执行，只有遇到转移指令（如，无条件转移、条件分支、调用和返回等指令）才改变指令执行的顺序。当执行到非转移指令时，CPU中的指令译码器通过对指令译码，知道正在执行的是一种顺序执行的指令，所以就直接通过对PC加“1”来使PC指向下一条顺序执行的指令；当执行到转移指令时，指令译码器知道正在执行的是一种转移指令，因而，控制运算器根据指令执行的结果进行相应的地址运算，把运算得到的转移目标地址送到PC中，使得执行的下一条指令为转移到的目标指令。

由此，可以看出指令在主存中的存放顺序是静态的，而指令的执行顺序是动态的。CPU能根据指令执行的结果动态改变程序的执行流程。

8. 主频越高，CPU的运算速度就越快吗?

答：CPU中的执行部件（定点运算部件、浮点运算部件）的每一步动作都要有相应的控制信号进行控制，这些控制信号何时发出、作用时间多长，都要有相应的时钟定时信号进行同步，CPU的主频就是同步时钟信号的频率。直观上来看，主频越高，每一步的动作就越快，CPU的运算速度也就越快。通常，同一类型处理器的平均的CPI（每条指令平均的时钟周期数）是固定的。所以，主频越快，一秒钟内执行的指令越多。例如，若CPI=2，则主频为500MHz的机器在一秒钟内执行10亿条指令；而主频为1GHz的机器在一秒钟内执行20亿条指令。 主频是反映CPU性能的重要指标，但只是反映了一个侧面，不是绝对的。如果一条指令所包含的动作分的很小，每一步动作所花的时间很短，因而，定时用的时钟周期很短，主频就高。此时执行一条指令所花的时间并没有缩短。如果不用流水线方式的话，CPU的运算速度并不会因为主频变高而变快。当然，现代计算机都采用流水线方式执行指令，使得每条指令大多能在一个时钟内完成，这样的话，主频变高，CPU的运算速度就变快了。

9. CPU中的控制器包含哪些基本部件?

答：CPU中的控制器包含以下基本的部件： 指令部件：包括程序计数器PC、指令寄存器IR、指令译码器等 时序部件：包括主频脉冲源、启停控制逻辑等 微操作信号（控制信号）生成部件：根据指令译码结果、当前机器状态、运行结果标志、时序信号等，用组合逻辑电路或微程序设计方式得到控制信号。有两大类控制信号：CPU内部控制信号；发到系统总线上的控制信号。 中断控制逻辑：包括中断允许触发器、中断查询、中断回答等控制逻辑

10. 流水线方式下，如何确定流水段的个数?

答：流水线方式下，一条指令的执行过程被分成了若干个操作子过程。由于每条指令所完成的功能不同，所包含的操作过程就不同。有的指令完成寄存器的内容的传送；有的是简单的加/减运算；还有的是复杂的乘/除运算。这些操作所花的时间相差很大，所以，这些指令如果都在同一个流水线中执行的话，就必须按最复杂的指令来设计流水线的流水段个数。 现代计算机一般把复杂度相近的指令用同一条流水线完成，而把复杂度相差很大的指令安排在不同的流水线中。

11. 流水线方式执行指令时，一条指令的执行时间变短了吗?

答：没有。流水线方式下，一条指令的执行过程被分成了若干个操作子过程。每个子过程由独立的功能部件来完成，以最复杂的子过程所花的时间为准设计时钟周期。这样，使得所有功能部件可以同时执行不同指令的不同子过程中的操作。理想情况下，经过若干周期后，流水线能在每个周期内执行完一条指令。但是，对于每条指令来说，它还是要经过若干子过程才能完成，所以一条指令的执行时间并没有变短。

12. 流水线方式执行指令时，总能在一个时钟内完成一条指令的执行吗?

答：不能。理想情况下，经过若干周期后，能在每个周期内执行完一条指令，即CPI=1。但是，当程序中出现以下情况时，流水线被破坏，因而，不能达到CPI=1。(1) 当有多条指令的不同阶段都要用到同一个功能部件时（资源冲突），后面指令要延时执行；（2）当程序的执行流程发生改变时（控制相关），原来按顺序取出的指令无效；（3）当后面指令的操作数是前面指令的运行结果时（数据相关），后面指令要延时执行。

13. 是否是流水段越多，指令执行越快? 为什么?

答：不是流水段越多，指令执行越快，因为： ① 流水段缓冲之间的额外开销增大 每个流水段有一些额外开销用在缓冲间传送数据、进行各种准备和发送等功能，这些开销加长了一条指令的整个执行时间，当指令间在逻辑上相互依赖时，开销更大。 ② 流水段间控制逻辑变多、变复杂 用于流水线优化和存储器(或寄存器)冲突处理的控制逻辑将随流水段的增加而大量增多，这可能导致用于流水段之间控制的逻辑比段本身的控制逻辑更复杂。

14. 除了有外部设备或他机向CPU发中断请求外，还有哪些情况会中断CPU正在运行的程序，转到其他相应的处理过程去执行?

答：以下四种情况发生时，会中断CPU正在运行的程序，转到其他相应的处理过程去执行。 （1）在程序执行过程中，若外设完成任务或发生某些特殊事件（如：打印机缺纸、定时采样计数时间到、键盘缓冲满等），会向CPU发中断请求，要求CPU对这些情况进行处理。处理完后，回到原被中断的断点处继续执行。这种情况，称为I/O中断，或外中断，特指由CPU外部的设备向CPU发的中断请求。 （2）在执行某条指令时，可能发生一些特殊的“异常事件”，如：缺页、溢出、除数为0、非法操作码等，使当前指令无法继续执行。此时也要求CPU中止原程序的执行，转到处理相应情况的程序去执行，处理完后，再回到发生异常的指令继续执行。这种情况，称为失效或故障（fault），是由正在执行的指令产生的。 （3）还有一类是人为设定的事件，在程序中事先设定一条特殊的指令，通过执行这条特殊指令，自动中止正在执行的原程序，转到一个特定的内核管理程序去执行，执行完后，回到那条特殊指令后面的一条指令开始执行。称为自愿中断或自陷(Trap)。这条特殊的指令称为 “访管指令”（访问管理程序）或“自陷指令”（自动调入陷阱），如80x86中的指令“INT n”。
（4）还有一种情况，既不是外部设备发出，也不是指令本身产生，是在执行指令过程中发生了硬件故障，如，电源掉电，线路故障等，无法继续执行。这类异常是随机发生的，对引起异常的指令的确切位置无法确定，出现这类严重错误时，原程序无法继续执行，只好终止，而由中断服务程序重新启动操作系统。 通常把发生这些情况的事件或设备称为中断源，有些教科书或计算机系统把它们称为“异常”事件或简称异常。 综上所述，上述四种中断源（异常事件）分为两大类：第一种称为

外中断（有时简称为中断Interrupt），后面三种称为内中断（也称为程序性中断，或软中断），分别为故障(fault)、自陷(Trap)和终止(Abort)。内中断是在执行特定的指令时发生的，不需要通过外部中断请求线进行中断请求。

15. 任何时候都可以申请中断并马上得到响应吗?

答：何时发出中断请求是由外设接口中的中断产生逻辑决定的，不受CPU的限制，但何时响应中断与CPU执行指令过程有关。CPU总是在一条指令执行完、取下条指令之前去查询有无中断请求。如果此时是开中断状态，并有未被屏蔽的中断请求发生，则CPU自动执行一条隐指令，进入中断响应机器周期，完成关中断、保护断点、取中断向量三个操作。所以不是任何时候都马上响应中断的，中断响应的条件有三个：（1）CPU处于开中断状态（中断允许触发器EINT置“1”状态）；（2）至少有一个未被屏蔽的中断请求发生；（3）一条指令执行结束。

16. 为什么在响应中断的时候保存断点，而在处理中断的时候保存现场?

答：断点是中断返回时被中断程序继续执行处指令的地址（即：响应中断时PC的值）和当时的程序状态字PSW的内容，所以必须在进入中断处理前先保存到栈中，否则，当取来中断服务程序的首地址送PC后，原断点PC的值就被破坏了。而现场是被中断的原程序在断点处各个寄存器的值，只要在这些寄存器再被使用前保存到栈中就行了，一般在实际处理中断事件过程中可能要用到这些寄存器，所以在实际处理之前的准备阶段来保存现场（寄存器压栈），而在实际处理后的结束阶段再恢复现场（寄存器出栈）。

17. 单重中断和多重中断的区别是什么?

答：单重中断情况下，在中断处理整个过程中，不允许响应新的中断请求，其做法是在中断响应过程中关中断后的整个中断处理过程中都不开中断（一直使中断允许触发器置“0”），直到中断处理结束后，才开中断，然后中断返回到原断点。

多重中断系统中，如果在进行某个中断请求的处理过程中，又发生了新的中断请求，则可以中止正在进行的中断处理，转到新的中断请求的处理程序执行。因此，在中断处理过程中，应该开中断，允许响应新的中断请求。其做法是在实际处理中断事件前就开中断，而不是像单重中断那样在处理后才开中断。这样保证在实际中断处理过程中可以响应新的中断请求。

18. 向量中断、中断向量、向量地址三个概念是什么关系?

答：中断向量：每个中断源都有对应的处理程序，我们称这个处理程序为中断服务程序，其入口地址称为中断向量。所有中断的中断服务程序入口地址构成一个表，称为中断向量表；也有的机器把中断服务程序入口的跳转指令构成一张表，称为中断向量跳转表。

向量地址：中断向量表或中断向量跳转表中每个表项所在的内存地址或表项的索引值，称为向量地址或中断类型号。

向量中断：是指一种识别中断源的技术或方式。识别中断源的目的就是要找到中断源对应的中断服务程序的入口地址，即获得向量地址。采用向量中断进行中断源识别的做法如下：采用某种硬件排队线路（如：菊花链、并行判优等），对所有未被屏蔽的中断请求进行排队，选出优先级最高的中断源，然后对其编码，得到该中断源的编号（可以转换为向量地址，有些书中就称其为向量地址），通过总线将其取到CPU中，转换成向量地址，从而取出中断服务程序入口地址，或跳转到中断服务程序。还有一种是用程序（称为中断查询程序）进行中断源识别的软件方法。

19. 禁止中断和屏蔽中断是同一个概念吗?

答：它们是两个完全不相关的概念。

禁止中断就是关中断，即：使中断允许触发器置为“0”，此时，任何中断请求都得不到响应；屏蔽中断是多重中断系统中的一个概念，是指某个中断正在被处理的时候，如果有其他新的中断请求发生，是否允许响应新发生的中断。它反映了正在处理的中断与其他各个中断之间的处理优先级顺序，所以每个中断都有一个中断屏蔽字，其中的每一位对应一个中断的屏蔽位，为“1”则对应的中断不能被响应。响应某个中断后，就会把它的中断屏蔽字送到中断屏蔽字寄存器中，在中断排队前，其中的每一位和中断请求寄存器中的对应位进行与操作，因而，只有未被屏蔽的中断源进入排队线路，从而有可能得到响应。

20. 中断响应优先级和中断处理优先级一样吗?

答：这是两个不同的概念。中断响应优先级是由硬件排队线路或中断查询程序的查询顺序决定的，不可动态改变；而中断处理优先级可以由中断屏蔽字来改变，反映的是正在处理的中断是否比新发生的中断的处理优先级低（屏蔽位为“0”，对新中断开放），如果是的话，就中止正在处理的中断，转到新中断去处理，处理完后回到原被中止的中断继续处理。

21. 实现控制单元CU的方式有哪些？（增加这个问题）

答：实现控制单元CU的方式有两类：

（1）组合逻辑控制，也称为硬连线路控制。由基本的门电路组合实现，因此，大多采用PLA方式实现。这种方式实现的控制器的处理速度快，但电路庞杂，制造周期长，不灵活，可维护性差。

（2）微程序控制。仿照程序设计的方法编制每个机器指令对应的微程序，每个微程序由若干条微指令构成，各微指令包含若干条微命令。所有指令对应的微程序放在只读存储器中。当执行到某条指令时，取出对应的微程序中的各条微指令，译码产生对应的微命令，送到机器相应的地方，控制其动作。这个只读存储器称为控存CS。微程序控制方式下，控制单元的设计简单、指令添加容易（灵活）、可维护性好，但速度较慢。