引入指令流水的目的：使指令间能并行执行，提高指令的运行速度。

要求：

1. 任务是可分解的
2. 每一个被分解出来的步骤需要用到一个独立的物理单元来实现，并且该物理单元有独立的控制逻辑
3. 每一个物理单元在完成任务的时候应该在给定的时间内完成

能引起指令流水的冲突的有：

1. 资源冲突
2. 数据相关性
3. 跳转指令

CISC：复杂指令集的处理器

要减轻汇编语言程序员的负担，CPU提供的指令越多越复杂越好。

尽量提供更多更复杂的指令。

CPU的设计会更复杂。

RISC：精简指令集的处理器

把过于复杂的功能从CPU中剔除出去，把不必要的指令精减掉。把处理器中空出来的晶体管用于提高剩余指令的运行效率、增加CPU的寄存器、增强处理器的功能。

RISC的特点：

1. 提供的指令提供的指令的长度往往是固定长度的。CPU对指令进行译码的时候也会快很多，指令流水的效率也会提高。
2. 只能用load和store对内存进行读写，不能通过其他指令进行读写。简化了对内存的操作。执行计算的时候操作数只能位于CPU寄存器或者指令中，寻址方式减少了很多，只剩下立即寻址和寄存器直接寻址。只能先从内存load操作数到CPU寄存器再进行计算。由于要操作的操作数必须先存入寄存器才能进行运算，故能有效减少资源冲突，进而提高指令流水的效率。
3. CPU内寄存器更多了。
4. 指令少了，只有一些基本功能的指令，编译器更复杂了。
5. 绝大多数指令都可以在一个时钟周期内完成
6. 指令更多是通过硬件来实现的，速度远远快于用微程序来实现。
7. 参数的传递：操作数是通过片内（CPU内部）的寄存器来传递的，不需要多一次对内存的访问。

冯诺依曼架构：控制单元（Control Unit）、ALU（算术逻辑单元）、存储器、输入设备、输出设备

设计以ALU为中心，ALU的负荷很大。

CPU有70%的时间在对内存的操作，因为不管是指令还是数据都是在存储在内存中。

改进的架构的以存储器为中心的。只有涉及到算术运算或者逻辑运算才会启动ALU。

逻辑上将指令和数据分成两部分存储，物理上还是存储在一起的，因为它们共用同样的数据总线和地址总线。所以不能同时读取指令和数据。

Harvard架构：把存储器从物理上分成独立的两部分，分别存储指令和数据，把指令和数据从物理上分开。这两个部分与CPU有各自独立的总线连接。

可以解决资源冲突问题，解决指令和数据不能同时读取的问题。

总线：连接计算机内部各个部件或各个子系统的一组公共的通路。

硬件中断：INTR和NMI这两个引脚用来接收中断请求，高电平就是接收到中断请求。

NMI获得的中断是不可屏蔽中断，INTR接收的中断请求是可屏蔽中断，只要IF = 0就是屏蔽中断请求。

软件中断：由INT指令引发的中断，与外围设备没有关系的。

00H~1FH这32个中断是被BIOS占用的。

20H~3FH这32个中断是被DOS占用的。

INT 10H BIOS中断

已经固化在BIOS中。

实现了汇编语言程序员与显示器之间的沟通。

这里通过寄存器AH来传递参数。

功能：

1. 清屏

AH = 06（）或者AH = 07

AL = 00：整个页面都清掉

BH = 07：正常属性

CH = 00：起点（左上角）的行数

CL = 00：起点的列数

DH = 24：终点（右下角）的行数

DL = 79：终点的列数

1. 设置光标位置

AH = 02：设置光标

BH = 00：设置到第0页

DH = 20：光标的行的坐标

DL = 20：光标的列的坐标

AH = 03：获取当前光标的位置

BH = 00：第0页。

DH和DL分别返回了光标所在的行和列

CX返回了当前光标的形状。

1. 显示一个字符
2. 画线画图
3. 改变显示模式

传递一个ASCII码时，传递一个字节表示ASCII码，还有一个字节用来表示显示的模式。

1. 文本模式

25行80列字符。

1. 视频模式

AH = 00

AL = 视频模式

INT 21H DOS中断

功能：

1. 在显示器中显示一个字符串

AH = 09：显示字符串

DX = 字符串起始地址。

默认碰到第一个’$’时，结束输出，且’$’不会被输出。

AH = 02：显示一个字符

DL = 要显示的字符

1. 从键盘读入一个字符

AH = 01：接收从键盘输入的一个字符

输入的字符存放在AL中。

1. 从键盘读入一个字符串

AH = 0AH：读入一个字符串

DX = 字符串存放的首地址

默认接收的时候，回车结束输入，并且回车会被读入

DB 6, ?, 6DUP(0FFH)

第一个单元定义了最大接收6个字符（包含回车），其中输入了多少个字符会存放到第二个单元（就是那个?）中，从第三个单元开始才是真正的字符串。

如果输入超过了字符串的大小，则只会读取前面的字符，加上最后一个回车符。

1. 不带回显的输入

AH = 07

回车与换行：

回车（CR）：0DH

光标移动到这一行的首部

换行（LF）：0AH

光标向下移动一行

LABEL指令

可以使某个变量或者函数有多个名称。

name LABEL attribute

attribute可以是BYTE、WORD、DWORD、QWORD、DT

DATA1 LABEL BYTE

MOV AL, DATA1

每次读取一个字节，如果BYTE改成WORD，则每次读取一个字。

如上面的字符串：

DATA1 DW 6, ?, 6DUP(?)

DATA1+2才是字符串的第一个字符，可以用：

BUF\_COUNT