**数字逻辑与数字系统**

**第二次作业**

## 列举三种处理器的架构，并结合一个具体的处理器型号说明每个功能模块的功能。

处理器的指令集架构（ISA）包含了基本数据类型、指令集、寄存器、寻址模式、存储体系、中断、异常处理以及外部I/O等。指令集架构包含一系列的操作码，以及由处理器执行的基本命令。目前主流的处理器架构为x86架构和ARM架构，除此之外还有MIPS架构、PowerPC架构和RISC-V架构等。根据指令集的复杂度区分，架构可被分为复杂指令集（CISC）和精简指令集（RISC），复杂指令集的指令数目多且复杂，但一条指令可以执行多个操作。

1. x86架构：x86架构是主要由Intel和AMD公司设计和研发的一种CISC架构，广泛应用于PC和服务器的处理器中，如Intel Core i9。目前的处理器基本采取x86架构的64位拓展版本x64，以实现更好的性能。x86架构的优点是较高的性能和兼容性。
2. ARM架构：ARM架构是一种RISC架构，其特点是低成本、低功耗，因此被广泛用于移动设备或嵌入式系统，如手机中的处理器骁龙8 Gen 2便是利用了基于ARMv9架构的Cortex-A715 CPU核心。
3. RISC-V架构：RISC-V架构是一个开源的RISC架构，它允许任何人设计和制造基于RISC-V的芯片和软件而无需支付专利费。它的设计考虑了小型、快速、低功耗的需求，但并没有对特定的微架构进行过度设计。其优点在于开源和精简。

以x86架构的Intel Core i9为例，处理器的功能模块一般包括控制单元、运算单元和存储单元。

1. 控制单元包括指令寄存器IR、指令译码器ID和操作控制器OC等，作用为协调整个电脑有序工作。它根据用户预先编好的程序，依次从存储器中取出各条指令，放在指令寄存器中，通过指令译码确定应该进行什么操作，然后通过操作控制器按确定的时序向相应的部件发出微操作控制信号。
2. 运算单元执行算术运算、逻辑运算等，根据控制单元的指令而进行操作。
3. 存储单元包括缓存和寄存器组，用于暂时存放数据。寄存器可以减少访问内存的次数，从而提高工作速度。

参考文献：

[1]四大主流芯片架构（X86、ARM、RISC-V和MIPS）. https://blog.csdn.net/maopig/article/details/121324070

[2]指令集架构-维基百科. https://zh.wikipedia.org/wiki/指令集架构

[3]x86-维基百科. https://zh.wikipedia.org/wiki/X86

[4]ARM架构-维基百科. https://zh.wikipedia.org/wiki/ARM架构

[5]RISC-V-维基百科. https://zh.wikipedia.org/wiki/RISC-V

[6]ARM Cortex-A715-维基百科. https://zh.wikipedia.org/wiki/ARM\_Cortex-A715

[7]cpu的基本结构及其工作原理-知乎. https://zhuanlan.zhihu.com/p/245119254

## 介绍处理器中流水线的概念。结合某一具体处理器，说明处理器的流水线如何设计和实现的。

处理器中，流水线是程序在执行时多条指令重叠进行操作的一种准并行处理实现技术。以MIPS处理器的五级流水线为例，每条指令在流水线中需要经历五个步骤，分别为IF（从缓存中获取下一条指令）、RD（从寄存器中读取数据）、ALU（算术和逻辑运算）、MEM（读写数据高速缓存中的内存变量）和WB（将结果写入寄存器）。在每一级流水线后增加一个寄存器，这样就使得每级流水线可以同时处理不同的指令。例如，当ALU正在执行第一条指令时，RD可以执行第二条指令，IF可以执行第三条指令。利用并行的处理方式，流水线技术可以大大提升处理器处理指令的效率。当然，指令的并行执行可能会带来硬件资源冲突、数据读写冲突的问题，这被称之为流水线冒险，可以采取不同的有效措施来解决结构冒险、数据冒险、控制冒险等。

参考文献：

[1] 处理器中的流水线技术-知乎. https://zhuanlan.zhihu.com/p/109574885

[2] 处理器中的流水线技术-CSDN. https://blog.csdn.net/pankul/article/details/8769979

## 课后习题

##### 用反演法求下列函数的反函数。

##### 2.8 写出下列各式的对偶式。

##### 2.13 写出下列各式的最小项表达式及最大项表达式。

根据最小项表达式与最大项表达式之间的转换关系，有

##### 2.14 将下列函数展开为最小项之和。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 00 | 01 | 11 | 10 |
| 0 |  |  | **1** |  |
| 1 | **1** | **1** | **1** | **1** |

##### 2.18 用卡诺图化简下列各式为最简与或式。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 00 | 01 | 11 | 10 |
| 00 |  | **1** |  |  |
| 01 | **1** | **1** |  | **1** |
| 11 |  | **1** | **1** |  |
| 10 |  | **1** | **1** |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 00 | 01 | 11 | 10 |
| 0 |  | **1** | **1** |  |
| 1 |  |  | **1** |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 00 | 01 | 11 | 10 |
| 00 | **1** |  |  | **d** |
| 01 | **1** | **1** |  |  |
| 11 | **1** |  | **d** |  |
| 10 |  |  | **1** |  |

##### 2.20 用卡诺图化简下列各式为最简与或式。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 00 | 01 | 11 | 10 |
| 00 | **1** |  |  | **1** |
| 01 | **1** |  |  | **d** |
| 11 | **d** |  | **d** | **1** |
| 10 | **1** |  |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 00 | 01 | 11 | 10 |
| 0 |  | **1** | **1** |  |
| 1 |  | **1** | **1** | **1** |

##### 2.21 直接根据逻辑表达式，填写卡诺图并化简下列各式为最简与或式。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 00 | 01 | 11 | 10 |
| 00 |  |  |  |  |
| 01 | **1** | **1** | **1** | **1** |
| 11 | **1** | **1** | **1** | **1** |
| 10 |  |  |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 00 | 01 | 11 | 10 |
| 00 | **1** | **1** |  | **1** |
| 01 | **1** | **1** |  | **1** |
| 11 | **1** | **1** |  |  |
| 10 |  |  |  |  |

分别画出的卡诺图，取公共部分如图。则

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 00 | 01 | 11 | 10 |
| 0 | **1** | **1** | **1** | **1** |
| 1 |  |  | **1** |  |

分别画出的卡诺图，取公共部分，并取反，如图。则