# CPU 与主存间的数据交互

20121034 胡才郁

(计算机工程与科学学院)

**摘　要**　CPU 与主存是计算机系统中的重要组成部分，在计算机中的数据交互发挥重要作用。本文梳理了 CPU 与主存的数据交互过程，并且分析了计算机内部在数据处理、数据传递、数据存储三个阶段之中的流程与操作细节，并以此为线索展开相关结构组成的介绍。

**关键词**　数据流动；数据通路；CPU；主存

## 1 引言

CPU 与主存都是冯·诺伊曼体系结构中计算机系统的重要组成部分。CPU 是计算机中负责读取指令，对指令译码并执行指令的核心部件。在计算机体系结构中，CPU 是对计算机的所有硬件资源（如存储器、输入输出单元） 进行控制调配、执行通用运算的核心硬件单元。计算机系统中所有软件层的操作，最终都将通过指令集映射为 CPU 的操作。而主存是计算机与 CPU 进行沟通的桥梁，主存负责数据的交换，其作用是用于暂时存放 CPU 中的运算数据，以及与硬盘等外部存储器交换的数据。

本文以 CPU 与主存之间的交互为线索，将此过程中数据分解为数据处理、数据传递、数据存储三部分。依次对应于 CPU 指令周期内的数据流动 、CPU 与主存之间的数据通路、主存内部数据的储存。通过这条线索，梳理了计算机中数据处理与存储的原理。

## 2 CPU 中指令周期的数据流

数据流动的第一阶段为数据的处理，对于 CPU 而言，在一个指令周期之下，CPU 通过内部数据流动完成对操作的控制。

计算机之中的数据流动依赖于 CPU 的加工处理与控制。而数据的处理依赖于 CPU 对于指令的执行。根据指令要求一次访问的数据序列，在每一条指令执行的不同阶段是不同的。并且，对于不同的指令，它们的数据流往往也不同。

CPU 从主存中取出并执行一条指令的时间称为指令周期。指令周期一般分为取值周期、间址周期、执行周期、中断周期。对于不同的指令而言，不一定会有间址周期与中断周期，并且，执行周期的数据流往往不同。本文将依次梳理指令周期各个阶段的执行流程与数据流。

### 2.1 取指周期

取值周期内，CPU 内部执行流程如下：

1.PC 寄存器指明了接下来要执行的指令在主存之中的存放地址，将 PC 当中保存的地址信息送入 MAR。

2．CU 控制单元通过控制总线控制主存储器中进行读操作或者写操作，这取决于主存中留有的两个接口 R 与 W，分别控制写操作与读操作。当 R 控制接口输送 1 高电平信号，指明主存需要进行读操作 CU 通过控制总线向主存发出了读信号。

3. 此时，MAR 指明了当前要取出的指令存放在什么地址，此地址信息通过地址总线送入总存。读出这个地址信息的数据，就意味着读出了要执行的这条指令。之后，这条指令通过数据总线被送入数据寄存器 MDR 之中。并将 MDR 之中的内容送入指令寄存器 IR 之中。

4．CU 控制单元发出读命令，使 PC 的值自动加 1。

取指周期的数据流向如下：

① PC → MAR → 地址总线 → 主存
② CU 发出控制信号 → 控制总线 → 主存
③ 主存 → 数据总线 → MDR → IR
④/⑤ CU 发出读命令 → PC 内容加 1

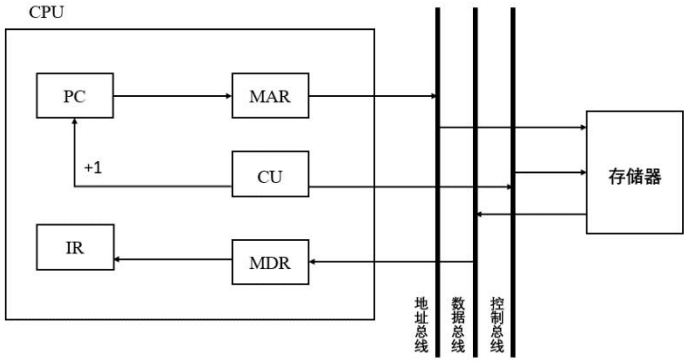


**图 1 一次取指周期的数据流示意图**

**表 1 取指周期的数据流**

| 步骤序号 | 说明 | 指令操作内容 |
|---|---|---|
| ① | 当前指令地址送至存储器地址寄存器 MAR | (PC) -> MAR |
| ② | CU 发出读控制信号，经控制总线传到主存 | 1 -> R |
| ③ | 将 MAR 所指主存中的内容经数据总线送入 MDR | M(MAR) -> MDR |
| ④ | 将 MDR 中的内容送入 IR | (MDR) -> IR |
| ⑤ | CU 发出控制信号，形成下一条指令地址 | (PC) + 1 -> PC |

## 2.2 间址周期

一条指令由操作码与地址码组成，对于地址码而言。既可以在指令中直接给出操作数的实际访存地址(有效地址)，也可以在指令的地址字段给出形式地址，将形式地址变换为有效地址再取操作数。

对于部分指令，指令的地址码采用了间接寻址的寻址方式，则这部分指令在指令周期中，有可能进入间址周期。

间址周期内，CPU 内部执行流程如下：

1．目前此指令的地址码只指明了有效地址的存放位置，因此，需要根据指令的地址码信息，把当前指令的地址码信息送入 MAR 之中。此操作有两种实现方式：(1)将 IR 中的地址信息送入 MAR 中 (2)将 MDR 中的地址信息送入 MAR 之中。由于在间址周期之前，已经进行了取指阶段，指令预先放置到了 MDR 之中，再将 MDR 中的指令进行复制，放置到 IR 之中。因此，此时 IR 与 MDR 中存储值相同，均为当前指令的信息。

2．CU 控制单元发出读信号 R，通过控制总线，启动主存做读操作。

3．依据 MAR 中存储的内容，将 MAR 所指主存中的内容经数据总线送至 MDR，读入相应的数据。读出的数据为有效地址 EA，此时找到了有效地址。

4．将有效地址送至指令的地址码字段。

间址周期的数据流向如下：

① Ad(IR)或 AD(MDR) → MAR → 地址总线 → 主存

② CU 发出控制信号 → 控制总线 → 主存

③ 主存 → 数据总线 → MDR(存放有效地址)

其中，Ad(IR)表示取出 IR 中存放的指令字的地址阶段。

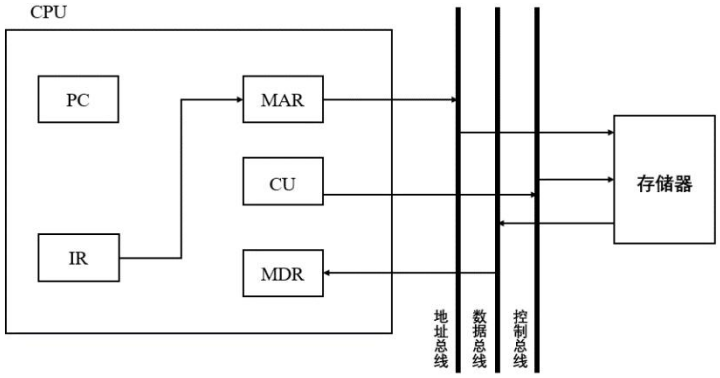


**图 2 一次间址周期的数据流示意图**

**表 2 间址周期的数据流**

| 步骤序号 | 说明 | 指令 |
|---|---|---|
| ① | 将指令的地址码送至存储器地址寄存器 MAR | Ad(IR) -> MAR<br>或 Ad(MDR) -> MAR |
| ② | CU 发出控制信号，启动主存做读操作 | 1 -> R |
| ③ | 将 MAR 所指主存中的内容经数据总线送入 MDR | M(MAR) -> MDR |
| ④ | 将有效地址送至指令的地址码字段 | (MDR) -> IR |

## 2.3 执行周期

执行周期的任务为根据 IR 中的指令字的操作码和操作数通过 ALU 操作产生相应的执行结果。由于此周期取决于指令具体功能，不同指令的执行周期操作不同，因此，并没有统一的数据流向。

## 2.4 中断周期

中断，是指暂停当前执行的程序，执行其它的程序。为了能够恢复当前的任务，需要保存当前任务执行处的断点。当 CPU 采用中断方式实现主机与 I/O 交换信息时，CPU 在每条指令执行阶段结束前，都要发送中断查询信号，以检测是否有某个 I/O 提出中断请求。如果有请求，CPU 则要进入中断响应阶段，又称中断周期。

一般使用堆栈来保存断点，在每一个程序运行时，操作系统都会为这个程序在主存之中开辟一段空间作为此进程的运行堆栈，使用堆栈指针 SP 指向当前运行堆栈的栈顶元素，进栈操作为先修改指针，后存入数据。

中断周期内，CPU 内部执行流程如下：

1．CU 控制单元将 SP 减 1，此时要将 PC 的值压入栈中，需要先将 SP 指针向低地址位置移动一个地址单元，并将 SP 指针指向的新位置存入 PC 的值。用这样的方式，记录下来当前程序运行的位置，修改后的

地址送入 MAR 之中，用来指明要将 PC 值存入哪一个主存单元之中。SP 指明了主存地址，将在此地址之中储存 PC 的值，需要把此地址信息存入 MAR 当中，用于指明接下来要进行写操作的是哪一个主存地址。

2．CU 通过控制总线，对主存发出写信号。

3. 此时需要将 PC 的值写入栈顶，而栈顶的地址信息已经存入了 MAR，根据 MAR 所指明的主存单元，则可得知需要写入的地址。而对于数据而言，一个数据想要写入主存，首先要将其放入 MDR，因此将 PC 中的数据放入 MDR 之中。

4．CU 控制单元通过控制总线将中断服务程序的第一条地址(入口地址)送入 PC。

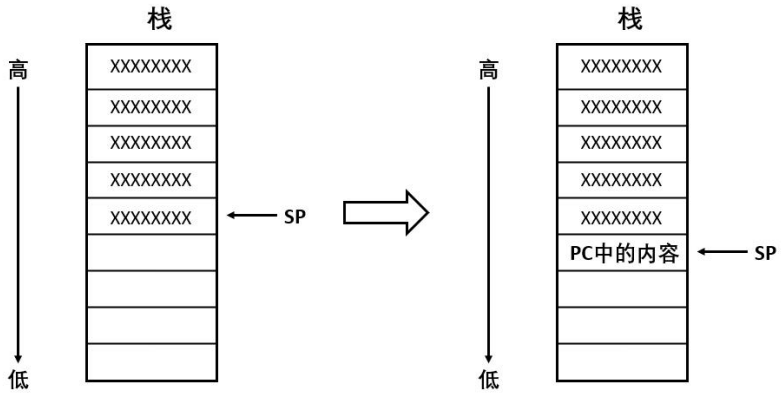


**图 3 堆栈寄存器中断点存储示意图**

中断周期的数据流向如下：

① CU 控制 SP 减 1, SP → MAR → 地址总线 → 主存

② CU 发出控制信号 → 控制总线 → 主存

③ 主存 → 数据总线 → MDR(存放有效地址)

其中，Ad(IR)表示取出 IR 中存放的指令字的地址阶段

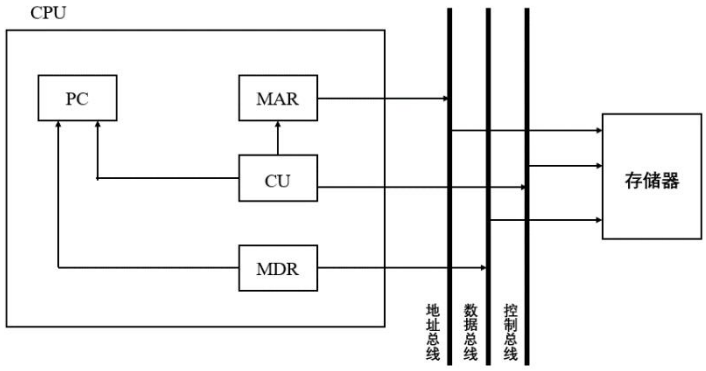


**图 4 一次中断周期的数据流示意图**

**表 3 中断周期的数据流**

| 步骤序号 | 说明 | 指令 |
|---|---|---|
| ① | CU 控制将 SP 减 1，修改后的地址送入 MAR | (SP)-1 -> SP, (SP) -> MAR |
| ② | CU 发出控制信号，启动主存做写操作 | 1 -> W |
| ③ | 将断点(PC 内容)送入 MDR | (PC) -> MDR |
| ④ | CU 控制将中断服务程序的入口地址送入 PC | 向量地址 -> PC |

# 3 数据通路

数据流动的第二阶段为数据的传递，CPU 与主存交互时依赖于内部总线。

## 3.1 数据通路的功能

数据在功能部件之间传送的路径称为数据通路。运算器与各寄存器之间的传输路径就是中央处理器内部数据通路。数据通路描述了信息从什么地方开始，中间经过哪个寄存器或多路开关，最后传送到哪个寄存器，这些都要加以控制。建立数据通路的任务是由“操作控制部件”来完成的，数据通路的功能是实现 CPU 内部的运算器与寄存器以及寄存器之间的数据交互。

## 3.2 数据通路的基本结构

### 3.2.1 CPU 内部单总线

将所有寄存器的输入端和输出端都连接到一条公共通路，这种结构比较简单，但数据传输存在较多的冲突现象，性能较低。当连接各部件的总线只有一条时，称为单总线结构；CPU 中有两条或更多的总线时，构成双总线结构或多总线结构。

### 3.2.2 CPU 内部三总线

将所有寄存器的输入端和输出端都连接到多条公共通路，相比之下单总线中一个始终只允许传一个数据，因而指令执行效率很低，因此采用多总线方式，同时在多个总线上传送不同的数据，提高效率。

### 3.2.3 专用数据通路方式

根据指令执行过程中的数据和地址的流动方向安排连接线路，避免使用共享的总线，性能较高，但硬件量大。

| 结构方式 | 物理实现 | 优点 | 缺点 |
|---|---|---|---|
| CPU 内部单总线 | 将所有寄存器的输入端和输出端都连接到一条公共通路上 | 结构简单易于实现 | 数据传输时存在较多的冲突现象 |
| CPU 内部三总线 | 将所有寄存器的输入端和输出端都连接到多条公共通路上 | 同时传送不同的数据 | 实现较难 |
| 专用数据通路 | 根据指令执行过程中的数据和地址的流动方向安排连接线路，避免使用共享的总线 | 性能较高 | 硬件量大 |

**表 4 数据通路结构对比**

## 3.3 主存与CPU之间的数据传送

主存与 CPU 之间的数据传送需要借助 CPU 内部总线完成。此处以 CPU 内部单总线结构，CPU 从主存读取指令为例，分析说明数据在数据通路中的传输过程，单总线结构如下图所示。

1. PC 指明了当前执行指令的地址。因此，将 PC 中的内容放置到总线上。此操作完成后，应当撤消总线上的控制信号。

2. 对主存进行读操作，因此 CU 控制单元应当通过控制总线对主存发出读信号，此时主存需要进行读操作，读操作的地址存放在 MAR 之中，此操作地址通过地址总线传送给主存。

3. 主存根据 MAR 中所指的地址读出相应的数据，再将此数据放入 MDR 之中，此数据的传送是通过数据总线。

4. 此时接通 MDRout 和 IRin 有效完成 MDR 与 IR 之间的数据传送。

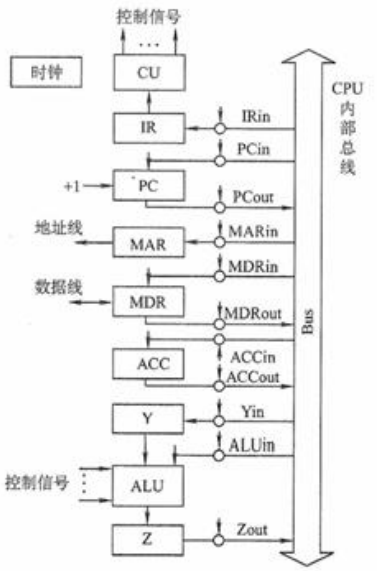


**图 5 主存与 CPU 之间的数据传送**

**表 5 数据通路结构对比**

| 步骤序号 | 说明 | 指令 |
|---|---|---|
| ① | PCout 和 MARin 有效，现行指令地址送入 MAR | (PC) -> BUS - > MAR |
| ② | CU 发出读命令 | 1 -> R |
| ③ | MDRin 有效 | MEM(MAR) -> MDR |
| ④ | MDRout 和 IRin 有效，现行指令送入 IR | MDR -> Bus -> IR |

# 4 主存储器对于数据的储存

数据流动的第三阶段为数据的存储，当 CPU 对于数据处理完成，并通过主线传送给主存后，主存进行对于数据的存储工作。

## 4.1 多级存储系统

为了解决存储系统大容量、高速度和低成本 3 个相互制约的矛盾，在计算机系统中，通常采用多级存储结构。存储系统层次结构主要体现在“Cache—主存”层次和“主存—辅存”层次。前者主要解决 CPU 和主存速度不匹配的问题，后者主要解决存储系统的容量问题。在存储体系中，Cache、主存能与 CPU 直接交换信息，辅存则要通过主存与 CPU 交换信息。而主存与 Cache、辅存都能交换信息。

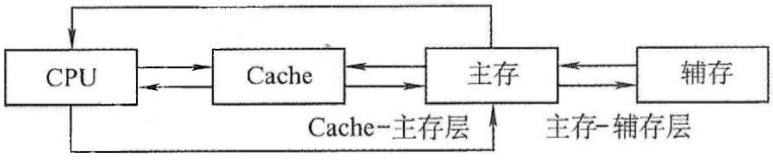


**图 6 三级存储系统的层次结构及其构成**

存储器层次结构的主要思想是上一层的存储器作为第一层存储器的高速缓存。从 CPU 的角度分析，“Cache—主存”层次速度接近于 Cache，容量与价位却接近于主存。从“主存—辅存”层次分析，其速度接近于主存，容量与价位却接近于辅存。这就解决了速度、容量、成本这三者之间的矛盾。

## 4.2 RAM

RAM 是随机存取存储器 (Random Access Memory) 的缩写，被用作计算机的短期存储器，用于放置其数据以方便访问。计算机拥有的 RAM 越多，它在给定时间可以存取的数据就越多。可以将 RAM 看作一个工作场所：一个巨大的工作台显然比一个小茶几更好操作。

主存储器有 DRAM 实现，靠近 CPU 的一层则由 SRAM 实现，他们都属于易失性存储器。

### 4.2.1 SRAM

通常把存放一个二进制位的物理器件称作存储元，它是存储器的最基本的构建。地址码相同的多个存储元构成一个存储单元，若干存储单元的集合构成存储体。

静态随机存储器(SRAM)的存储元是用双稳态触发器(六晶体管 MOS)来记忆信息的，因此，即使信息被读出后，它仍保持原状态而不需要再生(非破坏性读出)。

SRAM 的存取速度快，但集成度低，功耗较大。

### 4.2.2 DRAM

与 SRAM 的存储原理不同，动态随机存储器(DRAM)是利用存储元电路中栅极电容上的电荷来存储信息的，DRAM 的基本存储元通常只使用单个晶体管，因此，它比 SRAM 的密度要高很多。DRAM 采用地址复用技术，地址线是原来的 1/2，且地址信号分行、列两次传送。

相比于 SRAM 而言，DRAM 具有容易集成、价位低、容量大和功耗低等优点，但 SRAM 的存储速度比 SRAM 慢，一般用来组成大容量主存系统。

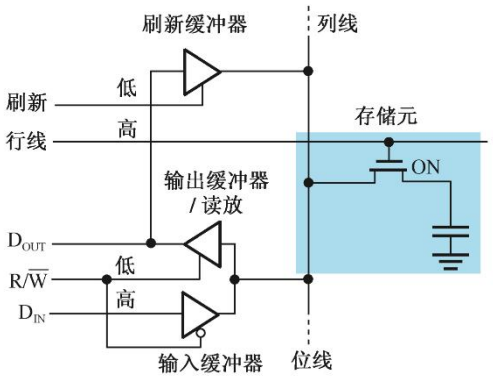


**图 7 单管 DRAM 存储元的工作原理**

**表 6 SRAM 与 DRAM 对比**

| 类型特点 | SRAM(静态 RAM) | DRAM(动态 RAM) |
|---|---|---|
| 存储信息 | 触发器 | 电容 |
| 破坏性读出 | 非 | 是 |
| 读出后需要重写 | 不需要 | 需要 |
| 运行速度 | 快 | 慢 |
| 集成度 | 低 | 高 |
| 发热量 | 大 | 小 |
| 存储成本 | 高 | 低 |
| 送行列地址 | 同时送 | 分为两次 (地址线复用) |
| 用途 | Cache | 主存 |

# 5 结语

计算机之中的数据离不开 CPU 与主存的数据交互，其中数据处理、数据传递、数据存储三个阶段分别对应于 CPU 指令周期内的数据流动、CPU 与主存之间的数据通路、主存内部的数据处理。通过分析数据在计算机中流动的过程，能够更好的理解计算机各个组成部分的工作原理。

**致谢** 十分感谢在百忙之中抽出时间审阅本文的老师。也正有着老师的带领，为我打开了计算机组成原理学习的大门。由于本人的学识和写作的水平有限，在本报告的写作中难免有僻陋，恳请老师多指教。

## 参 考 文 献

[1] 李梁. 内存数据管理与分析关键技术研究[D].东北大学,2020.DOI:10.27007/d.cnki.gdbeu.2020.000006.

[2] 冯平,尹家宇,宋长坤,余仕湖,李伯阳,陈铖颖,左石凯.非易失性静态随机存储器研究进展[J].半导体技术,2022,47(01):1-8+18.DOI:10.13290/j.cnki.bdtjs.2022.01.001.

[3] 计算机组成原理[M]. 科学出版社 , 白中英主编, 1994

# 附录 A 课程收获

## A.1 知识总结回顾

在本学期《计算机组成原理(1)》的学习之中，我们着重学习了计算机系统概述、数据的表示和运算、指令系统、中央处理器等知识。我了解了计算机系统层次结构、基本组成、指令格式、寻址方式，了解到了数制与编码、定点数、浮点数计算等，也学习了如何使用逻辑电路实现基本的原理。其中，除了本文重点分析的数据流动过程中 CPU 与主存的分工协调之外，流水线技术也十分吸引我。

计算机中的流水线是把一个重复的过程分解为若干个子过程，每个子过程与其他子过程并行进行。由于这种工作方式与工厂中的生产流水线十分相似，因此称为流水线技术从本质上讲，流水线技术是一种时间并行技术。

在流水线中按照时间周期的划分来分解问题的思路，也与本文中指令周期内的执行思路类似。

## A.2 遗憾与反思

冬季学期注定对我们来说是个难以忘记的学期，它有着我们正常线下上课的舒适节奏，有着正常考试周前复习的紧张刺激，也有着疫情突如其来的措手不及。所以令人十分遗憾的是在学习的过程中戛然而止，没有办法以考试的形式检验自己的所学成果。

由于学校三学期制度，使得我们的节奏进度较快，导致自己没有充足的时间去细细探究每一个计算机组成部件的设计思想、理念、应用的延申和拓展，让自己可以熟练掌握其应用，并且举一反三，甚至有机会动手亲自组建硬件设施。

在疫情隔离期间，我们有更多的时间去了解学业之外的事，比如在此期间我了解了很多系统结构的设计、汇编语言的学习、不同指令集的特点，极大地提高了我的兴趣。所以令我比较遗憾的是没有更早了解，这样或许可以更加提高自己冬季学期学习积极性。