1 计算机基础知识及性能评价

英语

1. *the very beginning* 开篇、肇始
2. *asynchronous* 异步的

UART 异步通信接口

3. *terminology* 用语、术语
4. *single-chip computer* 单片机

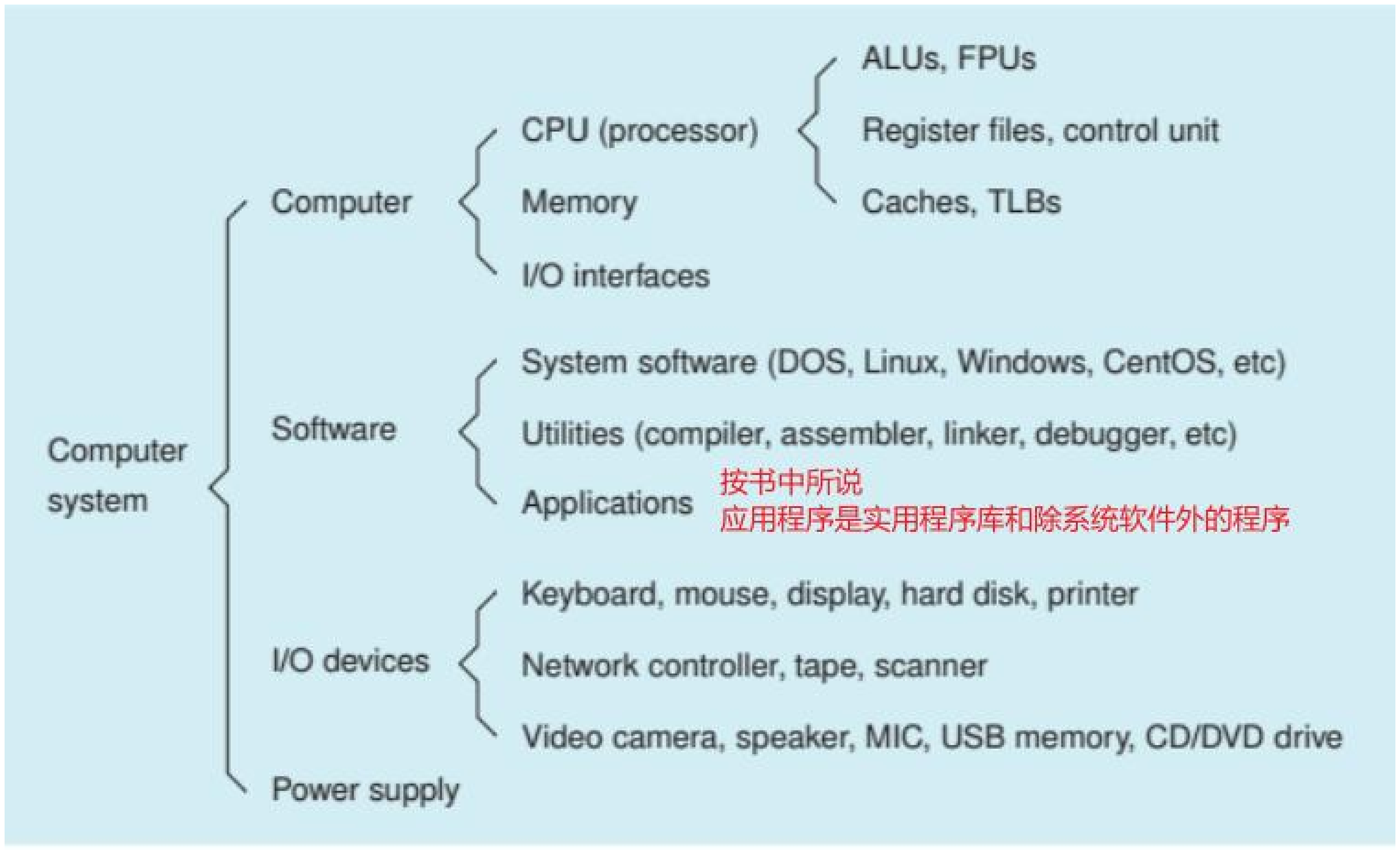
chip 芯片

5. *integrated* 集成的、综合的
6. *graphical* 图形的 *GUI*
7. *other than* 除……以外的
8. *peripheral* 外围的 *peripheral device* 外设
9. *fabricate* 编造、捏造
10. *vacuum* 真空
11. *arithmetic* 运算
12. *hence* 因此
13. *density* 密度
14. *investicate* 投资
15. prototype 原型
16. *by means of* 通过
17. *dedicated* 专用的
18. *simultaneously* 同时地
19. *incredibly* 不可思议地、
20. *come from* 来自
21. *benchmark* 基准
22. *bottleneck* 瓶颈
23. *eliminate* 消除
24. *symmetric* 相称的

1. *circuit* 电路、回路
2. *interconnection* 互联
3. *overview* 概述、概况
4. *transistor* 晶体管
5. *alternative* 可交换的
6. *magnetic* 磁性的
7. *instrument* 仪器
8. *corporation* 公司
9. *semiconductor* 半导体
10. *commercial* 商业的
11. *ubiquity* 普遍存在
12. *general-purpose* 通用的
13. *MIPS interlocked* 内部互锁的
14. *initiated from* 从……
15. *throw away* 扔掉
16. *denote* 表示
17. *overlap* 重叠
18. *intervene* 干预、介入
19. *corresponding* 相当的
20. *disturb* 打扰
21. *take place* 发生
22. *reciprocal* 相当的
23. *fed into* 输入
24. *utilization* 利用率
25. *distributed* 分布式的

1.1 计算机系统概述

1.1.1 计算机系统的组成



Computer system
Computer
CPU (processor)
ALUs, FPUs
Register files, control unit
Caches, TLBs
Memory
I/O interfaces
Software
System software (DOS, Linux, Windows, CentOS, etc)
Utilities (compiler, assembler, linker, debugger, etc)
Applications
按书中所说
应用程序是实用程序库和除系统软件外的程序
I/O devices
Keyboard, mouse, display, hard disk, printer
Network controller, tape, scanner
Video camera, speaker, MIC, USB memory, CD/DVD drive
Power supply

Figure 1.1 Computer system organization

任何一个包含“CPU/微处理器、存储器、外设接口”的集成电路芯片或者主板即被称为“计算机”

> *Generally, any IC chip or printed circuit board that contains these three kinds of components*注释1 *is called a* ***computer****.*

计算机系统包括上述的集成电路芯片/主板（计算机）、软件、外设和电源

> *The computer*注释2 *we often say is actually a “****computer system****”. A computer system consists of not only a computer, but also the software, I/O devices, and power supply*

因为计算机和外设均属于计算机硬件，所有可以称计算机系统包括“计算机硬件和计算机软件”（电源）

> *The computer and I/O devices belong to computer hardware. Therefore, we can say that a computer system consists of the computer hardware and the computer software (and a power supply).*

软件注释3

必要的软件是操作系统，作用是

> *It manages all the resources of the computer, I/O devices, and other software, and provides an interface for users to use the computer system.*

用于程序员开发可执行程序的编译器、编辑器、调试器以及其它一些库也是程序，有时称之为“**实用程序**”

> *The compiler, editor, debugger, and other libraries are also programs, sometimes we call them “**utilities**”*

所有除系统软件以外的程序+实用程序称之为应用

> *All programs other than the system software and utilities are called “**applications**”*

外设+外设接口

> *An I/O interface is a hardware controller that makes the communication between an I/O device and the CPU (and memory) possible*

I/O 接口是一个**硬件控制器，可实现 I/O 设备与 CPU（和内存）之间的通信**

Cache、TLB

> *Because the speed of the memory is much smaller than that of the CPU, in modern CPUs there is an instruction cache and a data cache. For the purpose of the fast virtual address translation, an instruction TLB and a data TLB are also fabricated in the CPUs*

Cache 主要是解决存储器和 CPU 速度之间的差异

TLB 主要是为了快速地进行虚拟地址的转换

1.1.2 计算机发展简史

电子计算机可以根据**构建计算机的技术分成四个阶段**

> *Electronic computers can be classified into four generations. Each generation used a new technology to build computers.*

第一代 1946-1955——真空管

第一代计算机（1946-1955 年）的主要特点是**使用真空管作为主要电子元件。这一代计算机主要使用机器语言，且只能执行单一任务，而且没有操作系统**

第二代 1956-1963——晶体管

第二代计算机（1956-1963 年）**使用晶体管作为处理元件，磁芯作为存储器**

汇编程序语言取代第一代计算机使用的机器语言，成为软件开发的主要工具

这一阶段**高级程序语言和操作系统**也得到发展

第三代 1964-1970——IC chip

第三代计算机（1964-1970 年）的特点是**使用集成电路芯片。这一阶段也将计算机体系结构和计算机实现分离开来**

> *The design made a clear distinction between architecture and implementation, allowing IBM to release a suite of compatible designs at dif- ferent prices*

摩尔定律：集成电路芯片上的晶体管数量大约每两年翻一番

> *The number of transistors on an IC chip has been doubling approximately every 2 years*

第四代 1971-至今——微处理器+半导体存储器

第四代计算机（1971 年至今）的主要特点是使用微处理器和半导体存储器

这一阶段集成数百万到数十亿的晶体管到一片集成电路芯片上。诞生了个人电脑

1.1.3 指令级 ISA

指令的构成

一条指令一定会包含操作码，此外也可能包含怎么得到操作数和结果存放在哪里

> *Generally, an instruction must consist of at least an operation code (opcode), which defines what will be done. Other parts that may be contained in an instruction include how to get the source operands and the place at which the execution result is stored.*

指令中的寄存器

CPU 内会存在一些寄存器。所有的寄存器都来自于寄存器文件，**每个寄存器文件中的寄存器都有一个独一无二的序列号。如果操作数是一个寄存器数据，那么就需要在指令中指明该寄存器的序列号**

> *Each register in the register file has a unique number. This register number is given in the instruction if a source operand is the register data*

指令所进行的操作

指令集操作的类型可以分为：

1. 整数运算和逻辑运算
2. 寄存器和存储器之间的数据传递
3. 条件分支和无条件跳转
4. 子程序调用和返回
5. 浮点计算
6. 输入/输出访问
7. 系统控制，如系统调用、中断返回、TLB 操作

> *The operation types of an ISA can be divided into the following:*
>
> *(i) integer arithmetic and logic calcula- tions;*
>
> *(ii) data movement between register file and memory;*

(iii) conditional branches and unconditional jumps;

(iv) subroutine call and return;

(v) calculations on floating-point numbers;

(vi) I/O accesses;

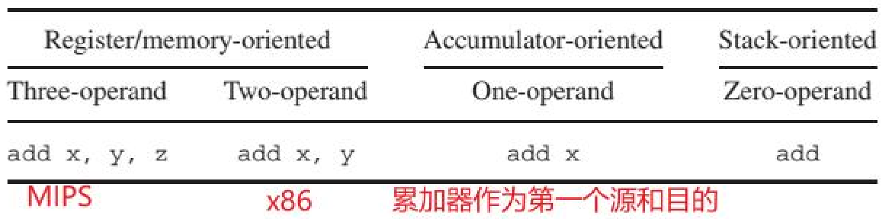
(vii) system controls, such as system calls, return from exceptions, and TLB manipulations.

指令集类型

四种 ISA 类型:

三操作数、二操作数、一操作数、零操作数

Table 1.1 Category of instruction set architecture



Register/memory-oriented		Accumulator-oriented	Stack-oriented
Three-operand	Two-operand	One-operand	Zero-operand
add x, y, z	add x, y	add x	add
MIPS	x86	累加器作为第一个源和目的	

> ***three-operand**: addu $6, $6, $4. This instruction adds the contents of the $6 and $4 registers, and places the sum in the $6 register*

***two-operand**: That is, addl %ecx, %ebx instruction adds the con- tents of the ecx and ebx registers, and places the sum in the ebx register.*

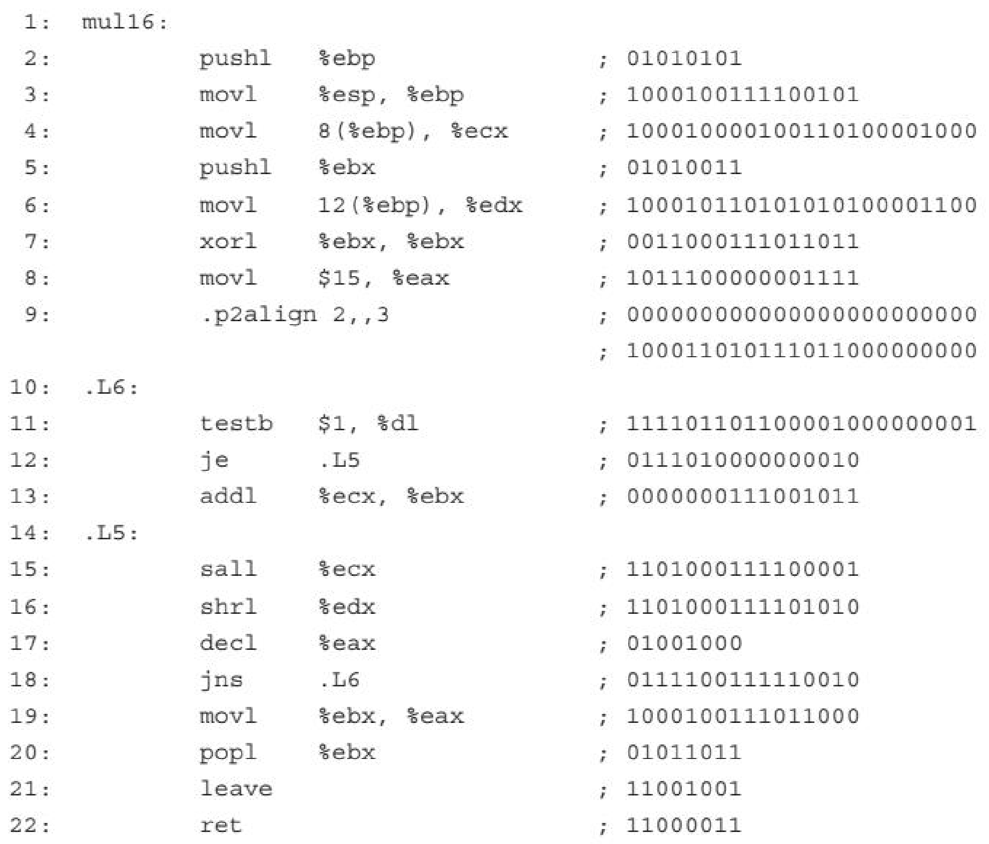
***one-operand**: In the CPUs that implement the one-operand ISAs, a special register, called an accumulator, acts as a default source register and the default destination register. It doesn' t appear in the instruction. The instruction needs only to give a register name or a memory address for the second source operand.*

***zero-operand**: Such ISAs are stack-oriented. The two source operands are popped from the top of the stack and the result is pushed onto the stack. The stack-top pointer is adjusted automatically according to the operation of the instruction.

The Bytecode, an ISA of JVM (Java virtual machine), is a typical example of the zero-operand ISAs.

MIPS 和 x86 ISA

x86

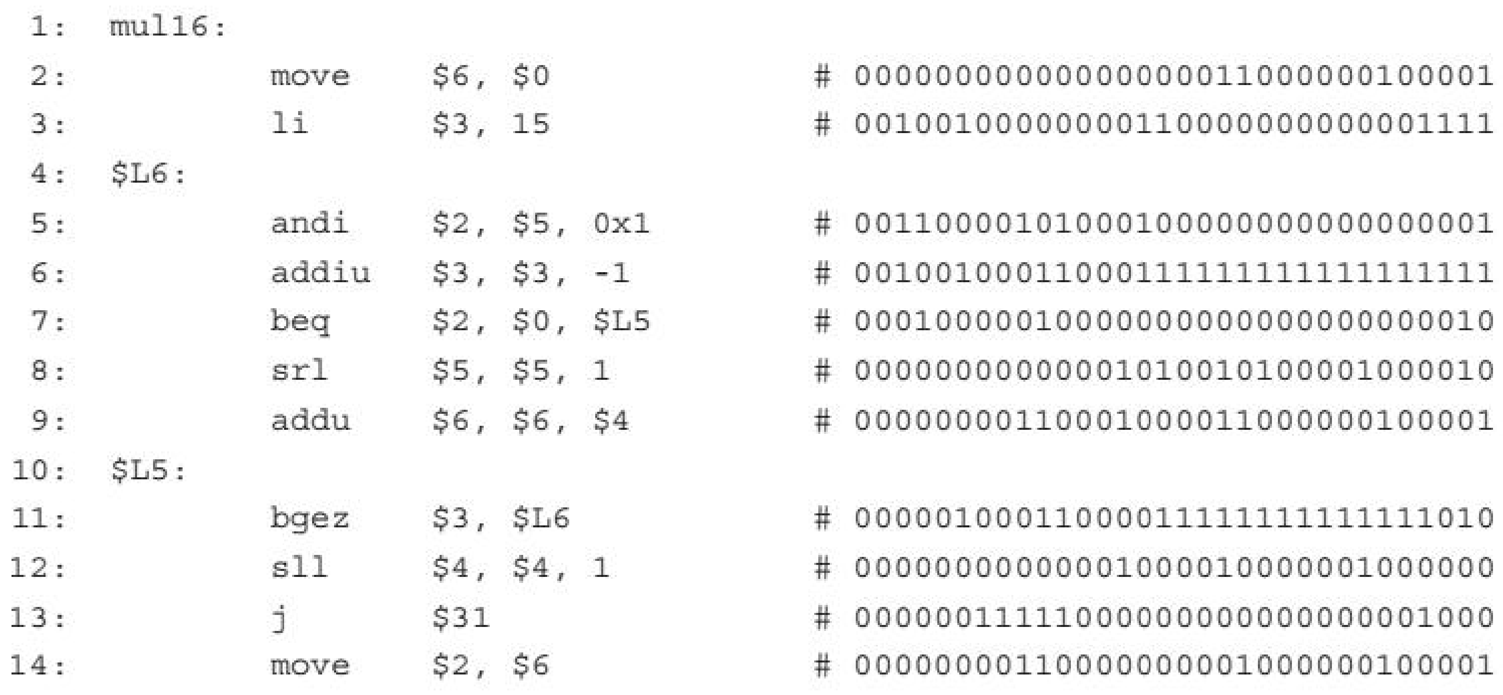


```
mul16:
        pushl   %ebp                ; 01010101
        movl    %esp, %ebp          ; 1000100111100101
        movl    8(%ebp), %ecx       ; 100010000100110100001000
        pushl   %ebx                ; 01010011
        movl    12(%ebp), %edx      ; 100010110101010100001100
        xorl    %ebx, %ebx          ; 0011000111011011
        movl    $15, %eax           ; 1011100000001111
        .p2align 2,,3               ; 000000000000000000000000
                                    ; 100011010111011000000000
.L6:
        testb   $1, %dl             ; 111101101100001000000001
        je      .L5                 ; 0111010000000010
        addl    %ecx, %ebx          ; 0000000111001011
.L5:
        sall    %ecx                ; 1101000111100001
        shrl    %edx                ; 1101000111101010
        decl    %eax                ; 01001000
        jns     .L6                 ; 0111100111110010
        movl    %ebx, %eax          ; 1000100111011000
        popl    %ebx                ; 01011011
        leave                       ; 11001001
        ret                         ; 11000011
```

1. 8 种寄存器：*In x86 ISA, there are eight registers: eax, ebx, ecx, edx, ebp, esp, esi, andedi*
2. 哈夫曼编码：*To shorten program codes, x86 uses a short opcode to encode the very commonly used instruc- tions*
3. 指令长度不固定：*From this we can see that the length of the instruction encodings is not fixed*
4. 不对当前指令译码无法知道下一条指令的起始地址→设计流水线困难：*If we do not decode the current instruction, we cannot know from where the next instruction starts. This feature makes the design of a pipelined x86 CPU difficult.*

MIPS

MIPS 全称“Microprocessor without interlocked piped stages ”，是无内部互锁[注释4]流水线级的微处理器



```
mul16:
          move    $6, $0          # 00000000000000000011000000100001
          li      $3, 15          # 00100100000000110000000000001111
$L6:
          andi    $2, $5, 0x1     # 00110000101000100000000000000001
          addiu   $3, $3, -1      # 00100100011000111111111111111111
          beq     $2, $0, $L5     # 00010000010000000000000000000010
          srl     $5, $5, 1       # 00000000000001010010100001000010
          addu    $6, $6, $4      # 00000000110001000011000000100001
$L5:
          bgez    $3, $L6         # 00000100011000011111111111111010
          sll     $4, $4, 1       # 00000000000001000010000001000000
          j       $31             # 00000011111000000000000000001000
          move    $2, $6          # 00000000110000000001000000100001
```

1. 32 个通用寄存器➞5 位的寄存器序列号：*MIPS32 ISA has a general-purpose register file that contains thirty-two 32-bit registers and hence a register number has five bits*
2. 指令长度固定➞易于设计流水： *the length of the MIPS instructions is fixed: All the instructions are represented with 32 bits. This feature makes the design of a pipelined MIPS CPU easy*

1.1.4 RISC 和 CISC

The MIPS belongs to RISC and the x86 belongs to CISC.

CISC

什么是复杂指令级集

复杂指令集即指令集中的**部分指令可以实现复杂的操作或者格式不统一**的指令集

> *An instruction set is said to be complex if there are some instructions that perform complex operations or the instruction formats are not uniform*

复杂指令集的作用

复杂指令集**可以提高代码密度**，使计算机系统可以使用少量的内存(包括缓存)来存储尽可能多的指令，**以降低成本和提高性能**

> *The CISC instruction set tries to enhance the code density so that a computer system can use a small amount of memory, including cache, to store as many instructions as possible for reducing the cost and improving the performance*

复杂指令集所做的措施

使一条指令执行尽可能多的操作➞微操作码

使每条指令的编码长度尽可能短➞可变长度的指令格式

> *CISC adopts two measures to reduce the code size – it lets an instruction perform as many operations as possible and makes the encoding of each instruction as short as possible. The first measure results in using microcode to implement the complex instructions, and the second measure results in a variable length of the instruction formats.*

CISC 中也存在简单形式的指令，存在**二八定律**

> *The analysis of the instruction mix generated by CISC compilers shows that about 80% of executed instruc- tions in a typical program uses only 20% of an instruction set and these instructions perform the simple operations and use only the simple addressing modes*

RISC

RISC 指令集特点

RISC CPU 有两个主要特点，如下：

1. 固定的指令长度➞使一个时钟周期取一条指令成为可能
2. *load-store* 机制➞使只有 *load/store* 指令可以从存储器中取数据至寄存器，其他指令的操作均只在寄存器上。使 RISC 指令集更简单

> *There are two main features in a RISC CPU. One is the fixed length of the instruction formats. This feature makes fetching an instruction in one clock cycle possible. The other feature is the so-called load/store architecture. It means that only the load and store instructions transfer data between the register file and memory, and other instructions perform operations on the register operands. This feature makes the operations of the RISC instructions simple. Both features make the design of the pipelined RISC CPUs easier than that of the CISC CPUs.*

目前的 RISC 和 CISC

RISC 是否能取代 CISC

不能，原因很简单：CISC 计算机的市场和软件资源

> *Was the CISC replaced by RISC? No. The reason is simple - the market. The Intel x86 ISA is widely used in the IBM-compatible PC, which is the most common computer system in the world. There is a huge amount of software resources that we cannot throw away.*

CISC 和 RISC 的交叉

1. CISC CPU 使用解码器去将 CISC 指令转变为 RISC 指令，然后使用 RISC 内核去执行指令
2. RISC CPU 也逐渐加入一些新的指令去支持复杂操作，比如多媒体操作

> *Today, most CISC CPUs use a decoder to convert CISC instructions into RISC instructions (micro-operations) and then use RISC cores to execute these instructions. Meanwhile, many RISC CPUs add more new instructions to support the complex operations, the multimedia operations for instance*

1.1.5 一些基本单位的意义

Table 1.2 Some base units

Powers of 2				Powers of 10					
Memory capacity				Clock frequency			Cycle length		
K	kilo	2^{10}	1,024	K	kilo	10^{3}	m	milli	10^{-3}
M	mega	2^{20}	1,048,576	M	mega	10^{6}	μ	micro	10^{-6}
G	giga	2^{30}	1,073,741,824	G	giga	10^{9}	n	nano	10^{-9}
T	tera	2^{40}	1,099,511,627,776	T	tera	10^{12}	p	pico	10^{-12}
P	peta	2^{50}	1,125,899,906,842,624	P	peta	10^{15}	f	femto	10^{-15}
E	exa	2^{60}	1,152,921,504,606,846,976	E	exa	10^{18}	a	atto	10^{-18}
Z	zetta	2^{70}	1,180,591,620,717,411,303,424	Z	zetta	10^{21}	z	zepto	10^{-21}
Y	yotta	2^{80}	1,208,925,819,614,629,174,706,176	Y	yotta	10^{24}	y	yocto	10^{-24}

存储：2 的幂，$K = 2^{10}, M = 2^{20}, G = 2^{30} dots$

频率：10 的幂，$K = 10^{3}, M = 10^{6}, G = 10^{9} dots$

1.2 计算机的基本结构

1.1 中已经提到，计算机由“CPU、存储、外设接口”组成。1.2 将对这三个组成部分详细介绍

1.2.1 RISC CPU 的基本结构

不带流水线的单周期 CPU

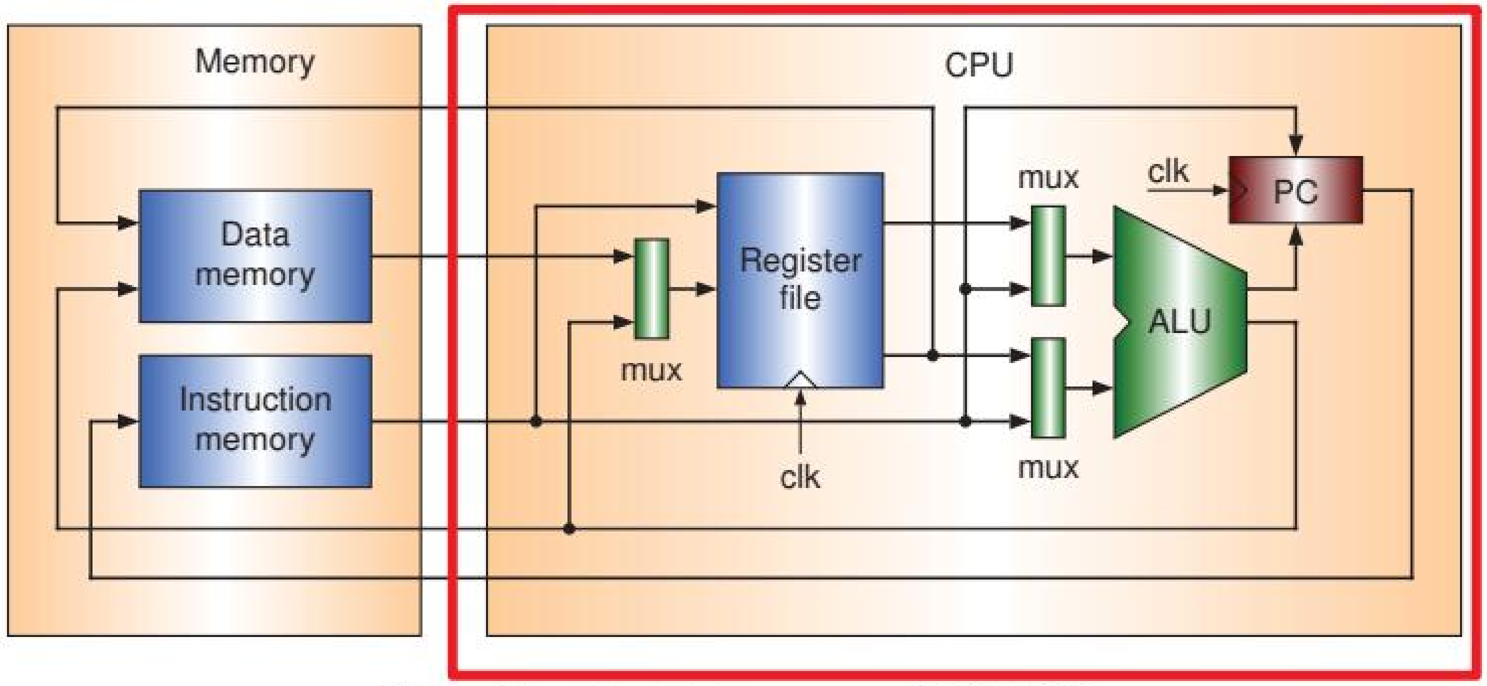


Figure 1.2 Simplified structure of RISC CPU

内部组件作用

1. PC 寄存器的内容是“当前执行指令的地址”，作为指令存储器的访存地址

   The instruction is fetched from the instruction memory. The content of PC (program counter) is used as the address of the instruction memory.

2. ALU 进行算术逻辑运算，ALU 的两端输入 A、B 均可以是寄存器内容也可以是指令中的立即数

   The ALU is responsible for calculations. Each of the two input operands of ALU can be either a register datum or an immediate provided in the instruction.

3. mux 根据选择信号，对多路输入进行一路输出选择

   The multiplexers (mux in the figure) are used for selecting an input from two inputs.

4. register file 中有 32 个寄存器，用于存储数据

   In the register file, there are a certain number of registers which can store data

5. ALU 的运算结果需要根据不同的指令作不同的处理。比如 load/store 指令，aluOut 是访存地址；普通运算指令，aluOut 是运算结果

6. 不同指令的处理

   1. load

      If the instruction is a load instruction, the data read from the data memory will be saved into the register file. In this case, the ALU output is used as the address of the data memory.

   2. store

      If the instruction is a store instruction, the data read from the register file will be saved into the data memory. In this case, the ALU output is used as the address of the data memory.

   3. conditional branch

      如果当前指令是条件分支指令，那么 aluOut 和运算的标志位即作为分支的判断条件。

      分支的目标地址是：$dest\ \ pc = cur\ \ pc +$ *指令立即数*

      If the instruction is a conditional branch instruction, the flags, outputs of ALU, are used to determine whether jumping to the target address or not. The target address of the branch can be calculated by adding an immediate to the current PC. If the branch is not taken, the PC + 4 is saved into PC for fetching the next instruction (a 32-bit instruction has four bytes and the PC holds the byte address of the instruction memory[注释5]).

五段流水线的 CPU

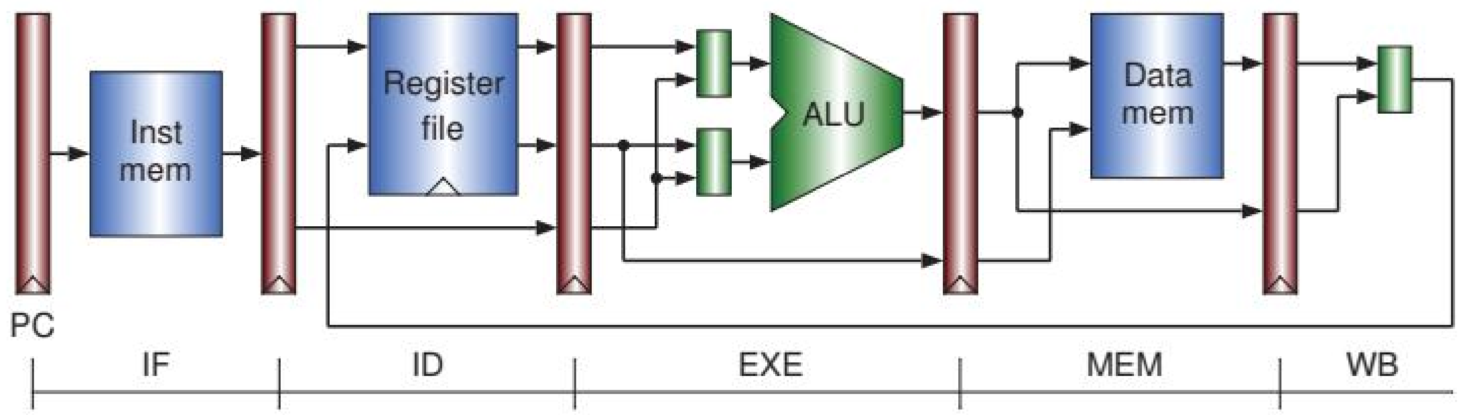


Figure 1.3 Simplified structure of pipelined CPU

五个阶段：IF、ID、EX、MEM、WB

段和段之间存在着流水线寄存器，用来传递数据

带 Cache 的 CPU

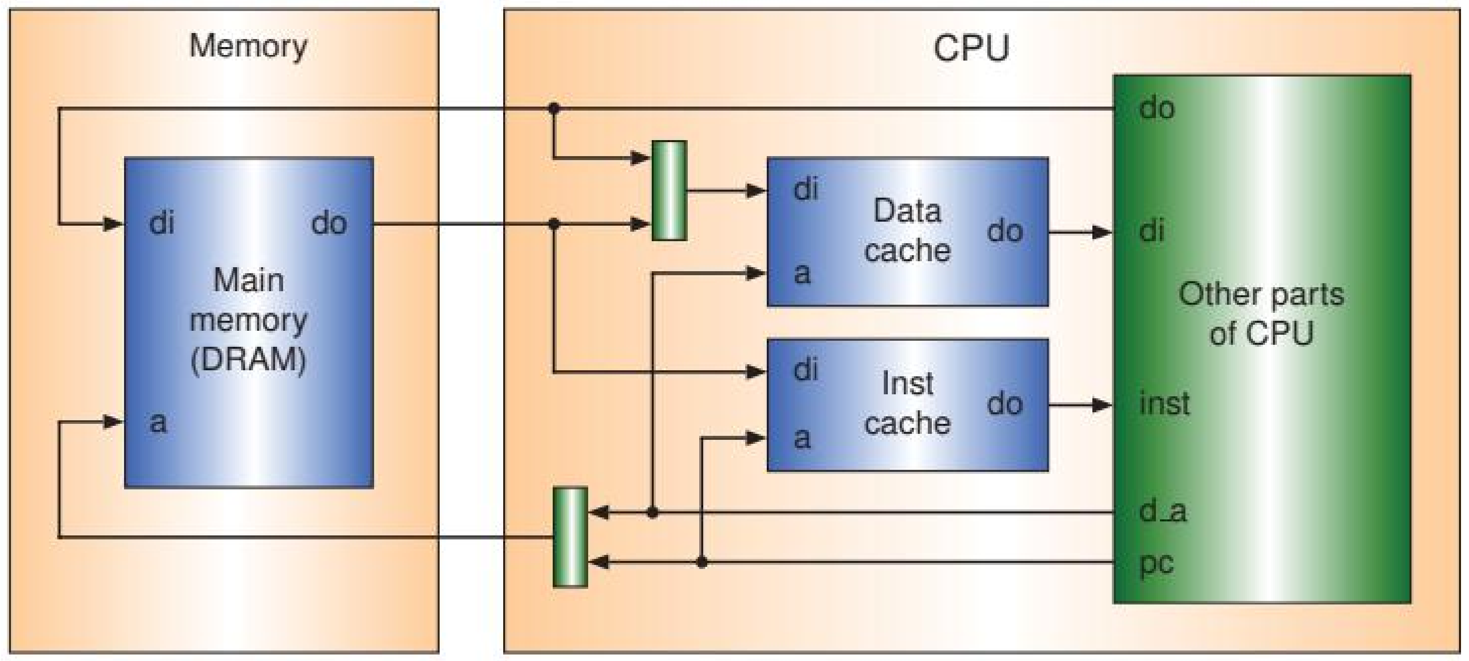


Figure 1.4 On-chip dedicated instruction cache and data cache

在之前的计算机发展历史中已经提到“从 1980s 起 CPU 的速度已经远快于主存的速度”，**为了减小主存和 CPU 之间的速度鸿沟，在 CPU chips 中构建了指令 cache 和数据 cache**

> *From early 1980s, CPUs have been rapidly increasing in speed, much faster than the main memory. Referring to Figure 1.4, to hide the performance gap between the CPU and memory, instruction cache and data cache are fabricated on the CPU chips.*

Cache 是一个小的快存，位于 CPI 和主存之间（实际在 CPU 内部），用于**存储来自最常用的主内存位置的数据或指令的副本**

> *A cache is a small amount of fast memory that is located in between the CPU and main memory and stores copies of the data or instructions from the most frequently used main memory locations.*

主存的实现从 DRAM➞SRAM。DRAM 便宜价格低速度慢，SRAM 昂贵价格高速度快

> *In modern computer systems, the main memory is commonly implemented with DRAM (dynamic ran- dom access memory). DRAMs have large capacity and are less expensive, but the memory control circuit is complex. Some high-performance computer systems use the SRAM (static random access memory) as the main memory. The SRAM is faster but more expensive than DRAM*

1.2.2 多线程 CPU 和多核 CPU

超标量 CPU

和流水线 CPU 每个时钟周期产生一个结果不同，超标量 CPU 通过在一个时钟周期内取指和执行多条指令，实现每个时钟周期产生多个结果。

但是由于**指令级并行性的缺陷——指令之间的控制和数据依赖**，超标量流水线每个时钟周期的平均执行指令条数仅为 1.2，相较于所为了实现超标量做出的硬件牺牲（寄存器重命名、保留站、乱序缓冲等），提升十分有限

> *The superscalar CPU tries to produce multiple results on every cycle by means of fetching and executing multiple instructions in a clock cycle. However, due to the control and data dependencies between instructions, the average number of instructions executed by a superscalar CPU per cycle is about 1.2. In order to achieve this 20% perfor- mance improvement, a superscalar must be designed with heavy extra circuits, such as register renaming, reservation stations, and reorder buffers, to support the parallel executions of multiple instructions. The superscalar CPUs cannot improve the performance further because of the lack of ILP (instruction level parallelism). Although the compilers can adopt the techniques of **the loop-unrolling and static instruction scheduling**, the improvement in performance is still limited.*

多线程 CPU

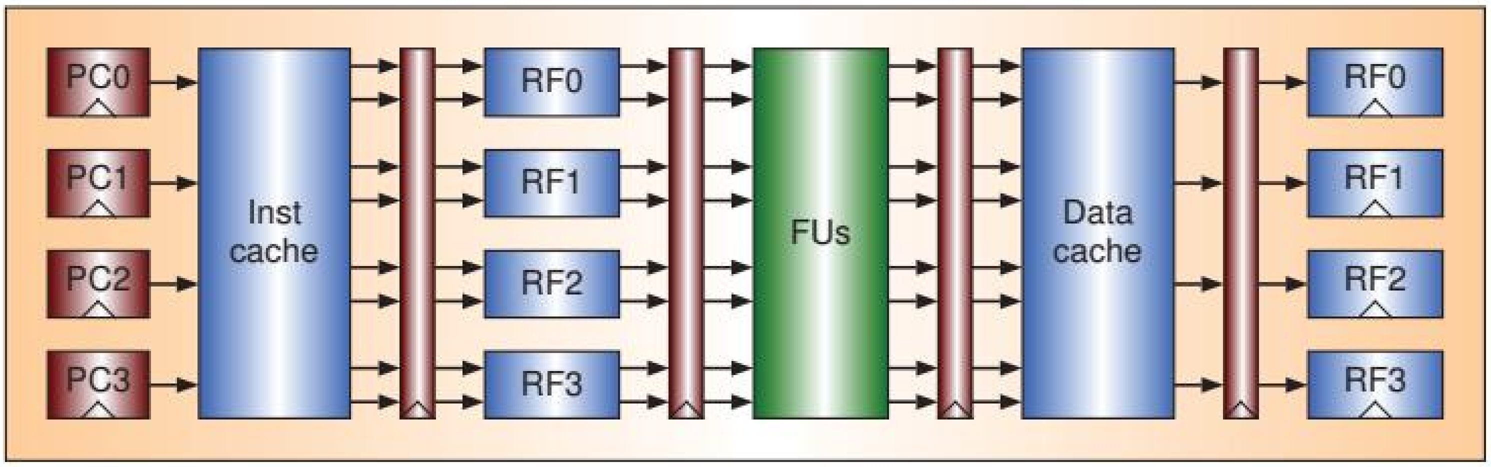


Figure 1.5 Simplified structure of a multithreading CPU

多线程 CPU 即并行执行多个线程

线程是一个指令的执行序列流。**每个线程都有一个 PC 和寄存器文件；各个线程之间共享指令 Cache、数据 Cache 和功能单元 FUs**。共享 FU 和缓存将提高这些组件的利用率，但会使控制变得复杂。

> *Each thread has a dedicated program counter and a register file, but the instruction cache, data cache, and functional units (FUs) are shared by all the threads. Sharing FUs and caches will increase the utilization of these components but make the control complex. Multithreading CPUs improve performance by exploiting the TLP (thread level parallelism).*

多核 CPU

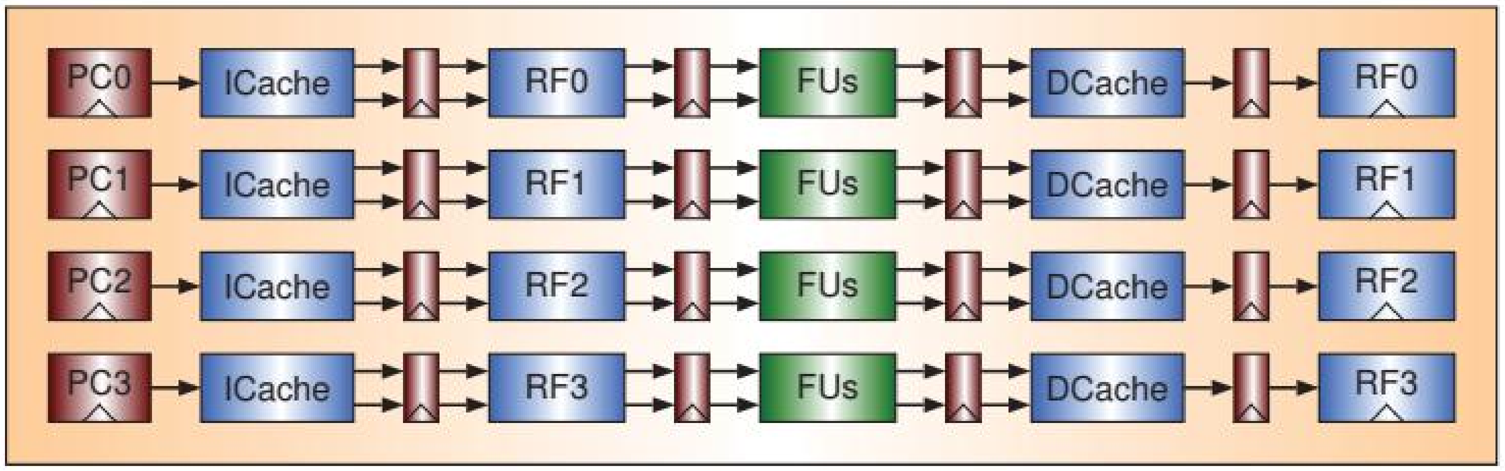


Figure 1.6 Simplified structure of a multicore CPU

多核 CPU 即一个集成芯片内部构建了多个普通的 CPU，这些普通的 CPU 可以是单周期 CPU、流水线 CPU、超标量 CPU 或者多线程 CPU

在多核 CPU 中，每个内核都有一个专用的 1 级 （L1） 指令缓存和一个数据缓存。所有内核可以共享一个二级 （L2） 缓存，或者每个内核都有一个专用的二级缓存。与多线程 CPU 相比，在多核 CPU 中，FU 和 L1 缓存不能由所有内核共享。这降低了组件的利用率，但使设计和实现变得简单

1.2.3 内存层次结构和虚拟内存管理

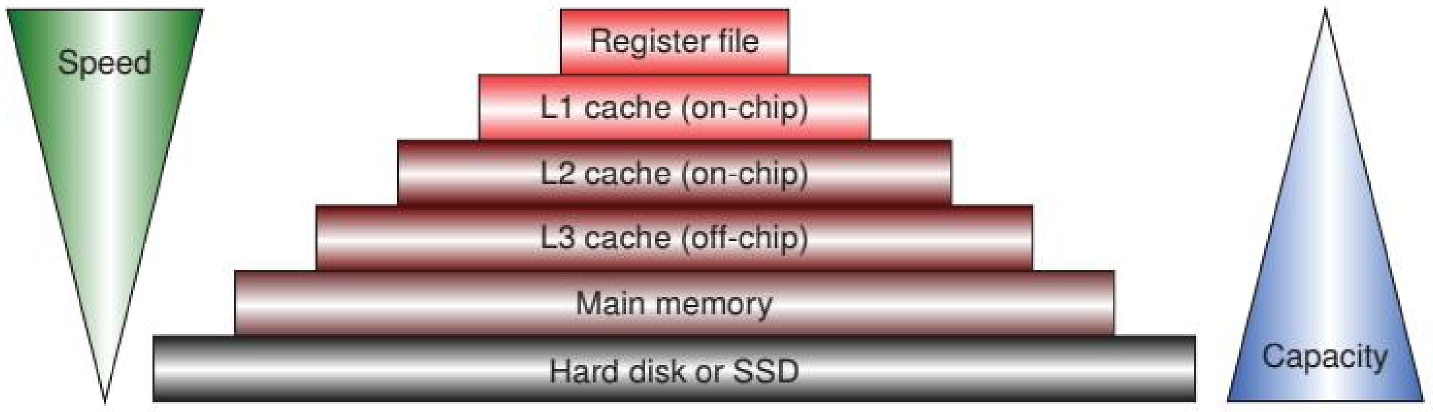


Figure 1.7 Memory hierarchy

内存层次结构

寄存器层

寄存器文件在 CPU 内部，是内存层次结构的最顶层，数量虽然少但是速度最快

> *The register file can be considered as the topmost level of the memory hierarchy. It has a small capacity but the highest speed.*

Cache 层

为了解决 CPU 和存储之间的速度差异，在 CPU 和存储之间插入了多级 Cache。在上图中有 3 级 Cache，on-chip Cache 的意思是板载 Cache，在 CPU 芯片内，off-chip Cache 是在 CPU 芯片外，由专用的 SRAM 芯片实现

> *There are three levels of caches (L1, L2, and L3). An on-chip cache is one that is fabricated in the CPU chip; an off-chip cache is one that is implemented with the dedicated SRAM chip(s) outside the CPU*

通常来说，L1 级 on-chip Cache 是采用指令和数据分离的技术，但 L2 级 on-chip Cache 是指令和数据共享存储

> *Commonly, the L1 on-chip cache consists of separated instruction cache and data cache, but the L2 on-chip cache is shared by instructions and data.*

磁盘层

磁盘位于内存层次结构的最底层，容量最大但速度很慢。目前 SSD 逐渐取代了磁盘（SSD 速度较之快）

> *The hard disk provides a large space for implementing the virtual memory. There is a trend that the SSD (solid-state drive) will replace the hard disk drive.*

虚拟内存管理

虚拟内存介绍

虚拟内存为要存储的正在运行的程序（进程）提供了较大的虚拟地址空间。**每个进程使用自己的虚拟地址空间，通常从 0 开始。虚拟地址空间的大小由程序计数器 PC 的位决定。***例如，32 位程序计数器可以访问 4 GB 的虚拟内存*

虚拟地址不能用于直接访问主内存，因为允许同时执行多个程序。**虚拟地址必须映射到物理内存位置**。以分页管理机制为例，在分页管理机制中，虚拟地址空间被划分为页——连续虚拟内存地址块。**因此，虚拟地址由虚拟页码和页面内的偏移量组成。只需将虚拟页码映射到物理页码。**

> *The virtual memory provides a large virtual address space for running programs (processes) to exist in. Each process uses its own virtual address space, usually starting from 0. The size of virtual address space is determined by the bits of the program counter. For example, a 32-bit program counter can access 4 GB of virtual memory. The virtual addresses cannot be used to access the main memory directly because multiple programs are allowed to be executed simultaneously. The virtual addresses must be mapped to physical memory locations. In a paging management mechanism, the virtual address space is divided into pages - blocks of contiguous virtual memory addresses. Thus, a virtual address consists of a virtual page number and an offset within a page. Only the virtual page number is needed to be mapped to a physical page number.*

虚拟地址到物理地址的映射是由**操作系统和 CPU 处理**的。每个进程都有一个页表，描述从虚拟地址到物理地址的变换

> *This mapping is handled by the operating system and the CPU. The operating system maintains a page table describing the mapping from virtual to physical pages for each process.*

TLB

如果操作系统每次对于进程提出的虚拟内存都要进行干预进行转换，那执行速度就会很慢。因此在现代 CPU 中，**构建了 TLB 来加快映射速度**，如下图

> *If the operating system had to intervene in every memory access, the execution performance would be slow. Therefore, in the modern CPUs, the TLBs are fabricated for speeding up the mapping,* as shown as below.

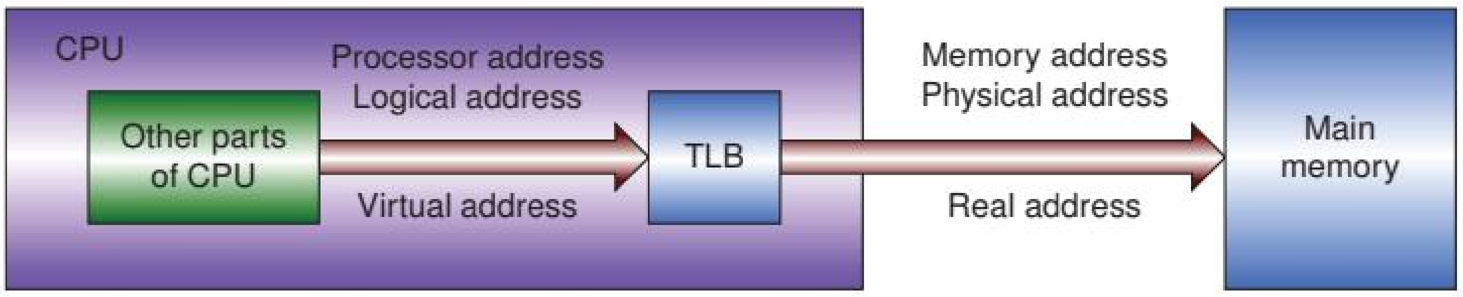


Figure 1.8 TLB maps virtual address to physical address

TLB 的组织结构与 Cache 非常相似。**TLB 存储最常用页表项的副本**。当 TLB 被命中时，可以立即获得物理页码，而不会干扰操作系统，因此实际映射速度非常快。**一般来说，虚拟地址也称为逻辑地址或处理器地址。物理地址也称为实际地址或内存地址**。请注意，对于特定的体系结构，这些名称可能具有不同的含义

> *The organization of the TLB is very similar to that of the cache. The TLB stores copies of the most frequently used page table entries. On a TLB hit, the physical page number can be obtained immediately without disturbing the operating system, so the actual mapping takes place very fast. Generally, the virtual address is also known as the logical address or processor address. And the physical address is also known as the real address or memory address. Note that for a particular architecture, these names may have different meanings.*

1.2.4 外设接口和总线

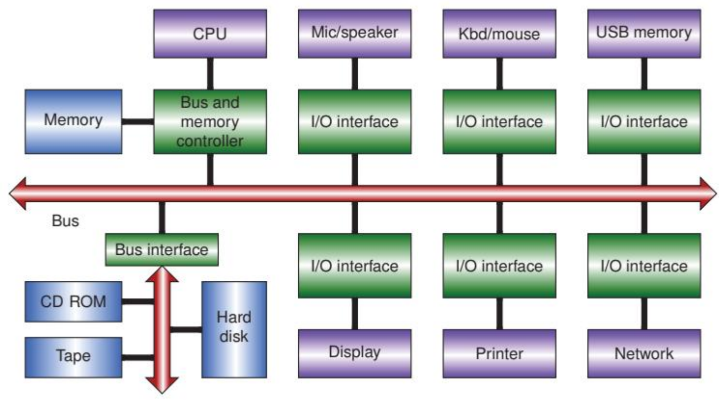


Figure 1.9 I/O interfaces in a computer system

如左图，在计算机系统中，计算机凭借 I/O 接口和外设进行交互。各个 I/O 接口通过总线相连

一个 I/O 端口对接口中的多个寄存器进行操作——写端口和读端口对应的寄存器不一样

> *An I/O interface may have several I/O ports, and each I/O port has a unique I/O address. The CPU can use the I/O address to write control information to a control register and to read state information from a state register in an I/O interface. Although these two registers are different, but their I/O addresses can be the same. This is a significantly different feature from the memory, where the read information from a memory location is definitely the same as the information that was written into the same location before reading.*

一个 I/O 接口可以有多个 I/O 端口，每个 I/O 端口都有一个独一无二的地址。CPU 可以使用这个 I/O 端口地址写控制信息到接口中的控制寄存器也可以从接口中的状态寄存器中读出状态信息，尽管控制寄存器和状态寄存器是不同的，但是**它们的 I/O 地址却可以相同**，这也是端口和内存不一样的地方——因为从内存位置读取的信息肯定与读取前写入同一位置的信息相同

I/O 端口地址有两种编址形式：统一编址和独立编址

统一编址：占用一部分内存的地址空间，和读写内存一样的方式去读写端口——*memory-mapped I/O address space*

eg:*Most of RISC CPUs use memory access instructions, load and store for example, to read and write I/O data.*

独立编址：有单独的编址空间，必须用特殊的 I/O 指令去读写端口——*dedicated I/O address space*

eg:*The Intel x86 has two special instructions, in and out, for these operations*

1.3 提高计算机的性能

1.3.1 计算机性能评价

执行时间计算公式

执行程序所用的时间计算公式如下：

$$Time = mathrm \times mathrm \times mathrm = \frac{mathrm \times mathrm}{mathrm}$$

其中，**I 是程序的指令数；CPI 是每条指令所用的平均时钟周期；TPC 是每个时钟周期的用时，值为 1/F**

> *where I is the number of executed instructions of a program, CPI is the average clock cycles per instruc- tion, and TPC is the time per clock cycle which is the reciprocal of the clock frequency (F).*

I、CPI、TPC 不是独立的。比如如果 RISC 处理器设计者想缩小 CPI 和 TPC 那么 I 就会相应的增加，CISC 处理器设计者想减少 I，那么必然会使 CPI 和 TPC 增加

> *Note that these three parameters are not independent. For example, CISC CPUs may reduce the instruc- tion count I by providing complex instructions, but it may result in an increase of CPI; RISC CPUs may decrease CPI and TPC, but it may cause the increase of I.*

这里介绍一下 IPC，IPC 是 CPI 的倒数，表示每个时钟周期能执行多少条指令，IPC 值一般越大越好

Amdahl 定律

Amdahl 定律用于**计算改进系统某一部分，对系统整体性能的提高**

> *When we calculate the expected improvement to an overall system when only part of the system is improved, we often use Amdahl's Law*

Amdahl 定律陈述了**一个计算机系统性能的提高是受限于这一系统能够改进的比例的**

> *Amdahl's Law states that the performance improvement to be gained from using some faster mode of execution is limited by the fraction of the time the faster mode can be used*

Amdahl 公式如下：

$$S = \frac{P_n}{P_o} = \frac{T_o}{T_n} = \frac{T_o}{T_o \times r/n + T_o \times (1-r)} = \frac{1}{r/n + (1-r)}$$

其中P_n表示被改进后的性能，P_o表示没有改进时的性能，T_o表示没有进行改进所执行程序的时间，T_n表示改进后执行程序所用的时间，r 是能够改进的比例，n 是所改进部分的优化量——优化 n 倍

> *Let Pn be the performance with enhancement, Po be the performance without enhancement, Tn be the execution time with enhancement, and To be the execution time without enhancement. r is the fraction enhanced and n is the speedup of the enhanced section.*

当 n➞∞时，可以得到$S = \frac{1}{1-r}$，从而可以画出 S 的瓶颈曲线，如下

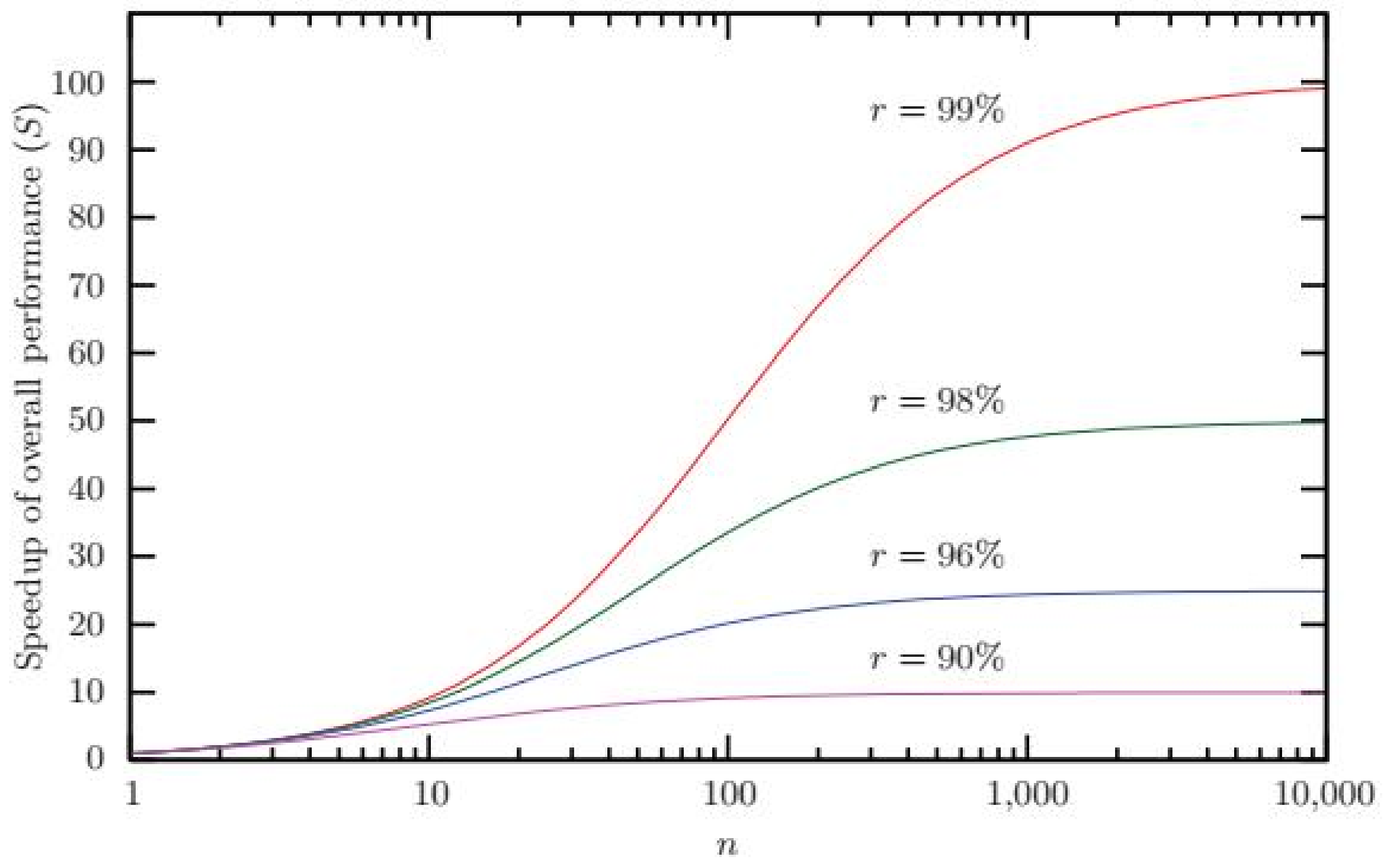


Figure 1.10 Amdahl's Law examples

也因此得到之前 Amdahl 定律所陈述的“Amdahl 定律陈述了一个计算机系统性能的提高是受限于这一系统能够改进的比例的”

> *Let n ➞ ∞;thenweget the upper bound of the overall speedup (S), which is 1/(1 − r). For example, if r = 50%, no matter how big n is, the overall speedup S cannot be larger than 2.*

1.3.2 追踪驱动仿真和执行驱动仿真

追踪驱动仿真

追踪驱动仿真是一种**根据输入轨迹中包含的指令和数据模拟计算机体系结构的行为**的用来估算**潜在计算机体系结构**性能的方法

> *Trace-driven simulation is a method for estimating the performance of potential computer architectures by simulating their behavior in response to the instruction and data references contained in an input trace*

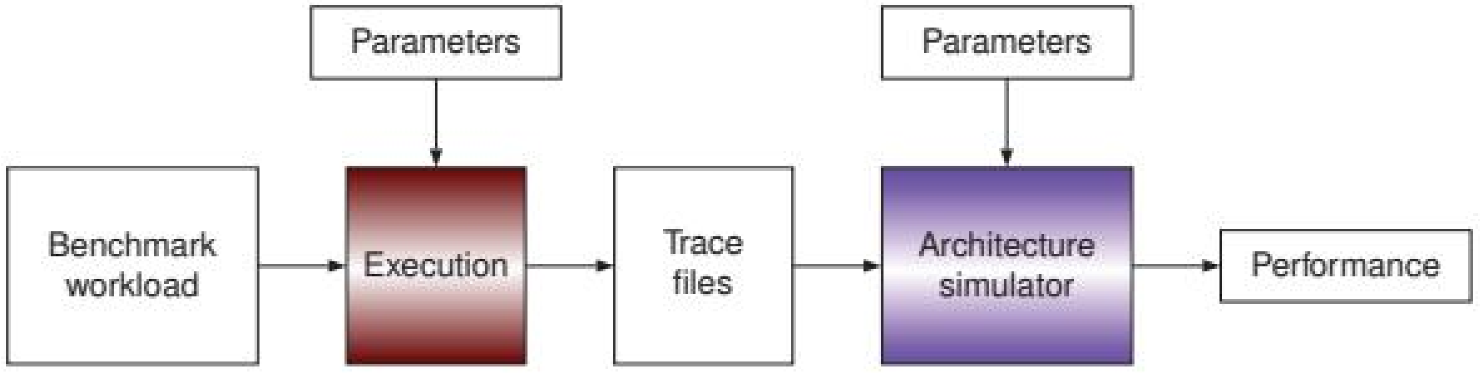


Figure 1.11 Trace-driven simulation

如上图所示，真实的机器执行一个基准程序，并写执行的指令信息（例如指令地址、操作码和数据地址）到一个追踪文件。然后将这个文件输入到一个用于性能研究的体系结构模拟器。**通过模拟器的仿真，我们可以知道机器的瓶颈性能并改变体系结构的配置来消除瓶颈**

> *As shown in Figure 1.11, a real machine is used to execute a benchmark program and write the executed instruction information, such as the instruction address, instruction opcode, and data reference address, to a trace file. This trace is then fed into an architecture simulator for the performance study, such as cache performance, ILP, accuracy of branch prediction, TLB performance, and the utilization of FUs.*

Through these simulations, we can find the performance bottleneck and change the architecture configurations to eliminate the bottleneck.

追踪驱动仿真的优点是：体系结构模拟器只要追踪文件的格式是已知的，那么就可以在任何机器上运行。但缺点是：需要保存大量的追踪文件

> *The architecture simulator can run any machine as long as the trace format is known, but the trace files are very large in general for real benchmark programs.*

执行驱动仿真

执行驱动仿真则是对追踪驱动仿真的优化，并不需要保存追踪文件，其流程如下图：

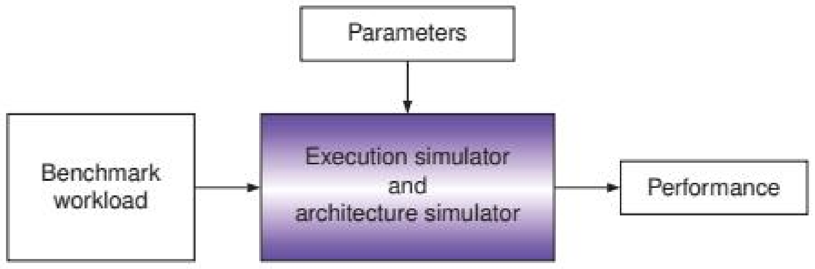


Figure 1.12 Execution-driven simulation

执行驱动仿真将“基准程序的执行”和“对追踪文件的仿真”合并，在执行的同时就运行性能研究模拟器，相比较于追踪驱动模拟来说，比较缓慢

执行驱动仿真和追踪驱动仿真的关系类似于“解释型语言”和“编译型语言”，前者需要边执行边仿真，后者是执行完毕再仿真，前者速度慢后者速度快

如果一个基准程序需要被使用很多次，采用追踪驱动仿真一次生成追踪文件是更好的选择

> *If a benchmark program will be used many times, it is better to generate the trace file once, and use the method of the trace-driven simulation.*

1.3.3 高性能计算机和内部互联网络

高性能计算机

高性能计算机可以分为**多处理器和多计算机**两种

> *High-performance computers can be categorized into multiprocessors and multicomputers*

多处理器 HPC

多处理器系统又叫做并行系统，多个 CPU 之间通过共享的内存来进行交流。这块共享的内存可以是集中的也可以是分布式的

> *In a multiprocessor system, there are multiple CPUs that communicate with each other through the shared memory. We also call it a parallel system. The shared memory can be centralized or distributed.*

超级计算机普遍采用分布式的共享存储的并行系统——DSM

> *Supercomputers are commonly parallel systems with the distributed shared memory (DSM).*

多计算机 HPC

多计算机系统又叫做分布式系统，多个计算机之间并不共享内存，一般通过消息传递（如 TCP/IP 协议）进行交流。

> *In a multicomputer system, there are multiple computers that communicate with each other via message-passing, over TCP/IP for example, and the memory in a computer is not shared by other computers. We call it a distributed system.*

在一些并行系统内，分布式的存储是通过消息传递机制来被各个处理器共享的

内部互联网络

1. remote memory

   A memory is said to be remote if the memory is not located on the same board with the CPU

2. locate memory

   A memory is said to be locate if the memory is located on the same board with the CPU

NUMA

CPU 可以直接访问 locate memory，但是访问 remote memory 则需要干预内部互联网络，因此访问 locate memory 是快于 remote memory 的。这种访问 locate memory 和 remote memory 速度差异的系统，称为“NUMA”—— nonuniform memory access

> *A CPU can access the local memory directly without disturbing the interconnection network. Access to the local memory is faster than access to the remote memory; we call this feature nonuniform memory access (NUMA)*

UMA

可以使用 SMPs（相称的多处理器）架构建立一个小的并行系统，在 SMP 系统内，**存储器单元通过总线和 CPUs 连接**，因此访问 remote memory 和 locate memory 速度相同，属于 uniform memory access——UMA 系统

> *When we want to build a small parallel system, a server for instance, we can use the architecture of symmetric multiprocessors (SMPs). In an SMP system, there is a common bus connecting CPUs and memory modules. The memory accesses in an SMP have uniform memory access (UMA).*

Amdahl 定律也适用于计算“使用超级计算机程序”的加速度

瓶颈仍是看可以改进的程序的比例

> *Amdahl’s Law is also suitable for calculating the speedup of a program using a supercomputer. If the sequential fraction of a program is 1%, the theoretical speedup using a supercomputer cannot exceed 100 no matter how many cores are used, as plotted for r = 99% in Figure 1.10.*

1.4 硬件描述语言 HDL

VHDL——*very high speed integrated circuit HDL*

习题

1. *What are the main differences between RISC and CISC?*

   指令长度和格式、寻址方式、load-store、寄存器、I/O 编址方式

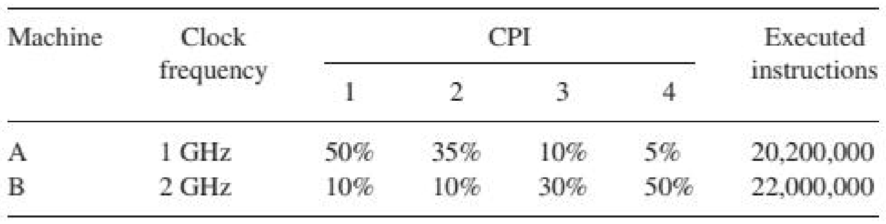
2. *Why is the microcode difficult to be pipelined?*

   微操作之间相互依赖难以划分流水段

   指令的执行时间不确定

   指令长度不确定

3. *Suppose that we have two machines: Machine A has a clock rate of 1 GHz and machine B has a clock rate of 2 GHz. We have made the measurements for these two machines as listed in the following table. Calculate the execution time and the MIPS (million instructions per second) of each machine.*



Machine	Clock frequency	CPI				Executed instructions
		1	2	3	4	
A	1 GHz	50%	35%	10%	5%	20,200,000
B	2 GHz	10%	10%	30%	50%	22,000,000

$$执行时间 = CPI \times I \times TPC = \frac{CPI \times I}{F}$$

$$MIPS = \frac{I}{执行时间 \times 10^6}$$

4. Let n = 10 and r = 75%. Calculate the overall speedup S by using Amdahl’s Law and give the upper bound of S.

$$S = \frac{1}{\frac{r}{n} + (1 - r)}$$

5. Let n = 1,000,000 and S = 500,000. Calculate the fractions r and 1 − r by using Amdahl’s Law. What are the meanings of these numbers when we compare the performance of a supercomputer to that of a uniprocessor computer?

超级计算机中 r 表示可并行化的计算部分，值为 0.999998，表示大部分都能实现并行处理

而单处理器计算机中，任务只能顺序执行，不能同时并行处理，0.000002 则表示大部分无法进行并行处理

- For a supercomputer:
  - The high value of r indicates that the supercomputer is well-equipped to leverage parallel processing and achieve remarkable performance improvements.
  - The minuscule value of 1 − r implies that even though there are sequential portions, they make up an extremely tiny part of the computation.
  - Supercomputers, with their multiple processing units, are designed to excel in workloads with high levels of parallelism.
- For a uniprocessor computer:
  - The low value of r would suggest limited potential for parallel speedup.
  - The value of 1 − r would be relatively higher, indicating a larger sequential portion that can't be efficiently parallelized.
  - Uniprocessor computers are inherently constrained in terms of parallel processing capabilities.

[注释 1] a CPU (or a microprocessor), a small amount of memory, and some I/O interface controllers

[注释 2] which can be used directly for an end user

[注释 3] Software is a collection of programs and related data

[注释 4] 利用了各种技术解决了流水线冲突导致的互锁

[注释 5] 因此 PC+4