中国科学技术大学
University of Science and Technology of China

COD HW1 & HW2

中国科学技术大学计算机学院
高能效智能计算实验室

2021年春

内容提要

- HW1
  - ✓1.14
  - ✓1.16
- HW2
  - ✓2.9
  - ✓2.24
  - ✓2.35
  - ✓2.40

习题 1.14

1.8 节引证了一个用性能公式的一个子集去计算性能的陷阱。为了说明它，下表是在不同的处理器中执行10^6 条指令序列的有关数据。

处理器	时钟频率	CPI
P1	4 GHz	1.25
P2	3 GHz	0.75

1.14.1 [5] <1.8> 一个常见的错误是，认为时钟频率最高的计算机具有最高的性能。这种说法正确吗？请用 P1 和 P2 来验证这一说法。

不正确　当P1和P2均执行10^6条指令时 假设P1与P2的指令系统相同

$T_{CPU} = \dfrac{IC_i \times CPI_i}{f}$，则有 $T_{CPU_1} = 3.125\times10^{-4}s$，$T_{CPU_2} = 2.5\times10^{-4}s$，$T_{CPU_1} > T_{CPU_2}$，即

P2 虽然时钟频率低，但拥有更高的性能

1.14.2 [10] <1.8> 另一个错误是，认为执行指令最多的处理器需要更多的 CPU 时间。考虑 P1 执行 10^6 条指令序列所需的时间，P1 和 P2 的 CPI 不变，计算一下 P2 用同样的时间可以执行多少条指令？

$n_2 = \dfrac{n_1 T_1}{T_2} = \dfrac{10^6 \times 3.125}{2.5} = 1.25\times10^6$，P2 用同样的时间可以执行$1.25\times10^6$ 条指令

运算速度

- 衡量计算机运算速度的指标，与许多因素有关，如机器主频、操作类型、主存速度等。
- 基准程序法(benchmark)
  - ✓ 考虑CPU、I/O结构、操作系统、编译器效率等
  - ✓ 评价计算机的实际工作能力
- 目前常用的测试程序：
  - ✓ 真实程序：C编译器、Matlab、Photoshop…(移植带来问题)
  - ✓ 核心程序：由从真实程序中提取的较短但很关键的代码构成。
    Livermore loops、Linpack
  - ✓ 合成测试程序（ Whetstone/Dhrystone ）
  - ✓ 测试程序组（测试程序包：Benchmarks）：选择一组各个方面有代表性的测试程序，组成的一个通用测试程序集合。

习题 1.14

1.8 节引证了一个用性能公式的一个子集去计算性能的陷阱。为了说明它，下表是在不同的处理器中执行10^6 条指令序列的有关数据。

处理器	时钟频率	CPI
P1	4 GHz	1.25
P2	3 GHz	0.75

1.14.3 [10] <1.8> 一个常见的错误是用 MIPS（每秒百万条指令）来比较 2 台不同的处理器的性能，并认为 MIPS 最大的处理器具有最高的性能。这种说法正确吗？请用 P1 和 P2 验证这一说法。

$$MIPS = \frac{n_{MIPS}}{T_{CPU} \times 10^6}$$

则对处理器 P1，$MIPS_1 = 3.2 \times 10^3 MIPS$

对程序 b，$MIPS_2 = 4 \times 10^3 MIPS$

如果两者没有相同的指令系统，只依据MIPS判断性能高低是没有意义的。

例如，p1和p2执行功能相同的程序，由于p1的指令集结构和编译技术更科学，在处理器p1中该程序的功能只需要1000条指令就能完成；而在p2中需要10000条指令才能完成相同的功能。

这种情况下p1显然有更高的性能，即使p2有着MIPS指标更高。

运算速度

存在问题

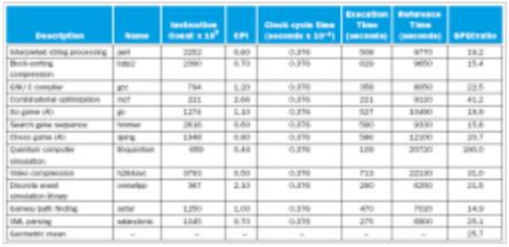
编辑

MIPS是指令执行的速率，规定了性能和执行时间成反比，越快的计算机具有越高的MIPS值。从表面看，MIPS既容易理解，又符合人的直觉。其实，用MIPS作为度量性能的指标存在三个问题。

首先，MIPS规定了指令执行的速率，但没有考虑指令的能力。我们没有办法用MIPS比较不同指令集的计算机，因为指令数肯定是不同的。

其次，在同一计算机上，不同的程序会有不同的MIPS，因而一台计算机不会只有一个MIPS值。例如，将执行时间用MIPS、CPI、时钟频率代入之后可得：

MIPS = 指令数/(执行时间 * 10^6) = 指令数 / (指令数 * CPI / 时钟频率 * 10^6) = 时钟频率 / (CPI * 10^6)



上图显示了SPEC2006在Intel Core i7上的CPI最大值和最小值是相差5倍的，MIPS也是如此。

- 早期衡量运算速度的普通方法：完成一次加法或乘法所需的时间
  - ✓CPI（Cycle per Instruction）
    - 指令周期，表示执行一条指令所需的平均时钟周期数

$$CPI = \frac{\text{执行程序所需的CPU时钟周期数}}{\text{程序包含的指令条数}}$$

  - ✓MIPS（Million Instruction Per Second）
    - 百万条指令每秒，即单位时间内执行的指令数

$$MIPS = \frac{\text{程序的指令条数}}{\text{程序的执行时间}\times 10^6}$$

  - ✓MFLOPS（Million Floating Point Operation Per Second）
    - 百万次浮点操作每秒，用来衡量机器浮点运算的性能

$$MFLOPS = \frac{\text{程序的浮点操作次数}}{\text{程序的执行时间}\times 10^6}$$

另一个常见的性能标志是 MFLOPS（每秒百万条浮点指令），其定义为 MFLOPS = 浮点操作的个数/执行时间 $\times 10^6$。它与 MIPS 有同样的问题。考虑下表所示的程序，在时钟频率为 3 GHz 的处理器上运行。

	指令数	读/写	浮点	分支	CPI（读/写）	CPI（浮点）	CPI（分支）
程序 a	10^6	50%	40%	10%	0.75	1	1.5
程序 b	3×10^6	40%	40%	20%	1.25	0.70	1.25

1.14.4 [10] <1.8> 求程序的 MFLOPS 值。

1.14.5 [10] <1.8> 求程序的 MIPS 值。

1.14.6 [10] <1.8> 求程序的性能，并与 MFLOPS 和 MIPS 值作比较。

1.14.4

$$T_{CPU} = \sum \frac{IC_i \times CPI_i}{f}$$

则对程序 a，$T_{CPUa} \approx 3.083\times10^{-4} s$，对程序 b，$T_{CPUb} = 1.03\times10^{-3} s$

又 $MFLOPS = \dfrac{n_{MFLOPS}}{T_{CPU}\times10^6}$

则对程序 a，$MFLOPS_a = 1.2973\times10^3\, MFLOPS$

对程序 b，$MFLOPS_b = 1.165\times10^3\, MFLOPS$

另一个常见的性能标志是 MFLOPS（每秒百万条浮点指令），其定义为 MFLOPS = 浮点操作的个数/执行时间 $\times 10^6$。它与 MIPS 有同样的问题。考虑下表所示的程序，在时钟频率为 3 GHz 的处理器上运行。

	指令数	读/写	浮点	分支	CPI（读/写）	CPI（浮点）	CPI（分支）
程序 a	10^6	50%	40%	10%	0.75	1	1.5
程序 b	3×10^6	40%	40%	20%	1.25	0.70	1.25

1.14.4 [10] <1.8> 求程序的 MFLOPS 值。

1.14.5 [10] <1.8> 求程序的 MIPS 值。

1.14.6 [10] <1.8> 求程序的性能，并与 MFLOPS 和 MIPS 值作比较。

1.14.5

$$MIPS = \frac{n_{MIPS}}{T_{CPU}\times 10^6}$$

则对程序 a，$MIPS_a = 3.24\times10^3 MIPS$

对程序 b，$MIPS_b = 2.9126\times10^3 MIPS$

1.14.6

由 1.14.4 知对程序 a，$T_{CPUa} \approx 3.083\times10^{-4} s$，对程序 b，$T_{CPUb} = 1.03\times10^{-3} s$

程序 a 的性能要优于程序 b，使用 MIPS 与 MFLOPS 得出的结果同样是程序 a 更好

$Performance = 1/T \quad Performance_a = 3.243\times10^3; Performance_b = 0.971\times10^3$

还有一个易犯的错误是有关在多处理器系统中运行，希望只改进一部分例行程序来改进整体性能。下表表示某个程序的 5 个例程在不同数目处理器中的执行时间。

	处理器数	例程 A（ms）	例程 B（ms）	例程 C（ms）	例程 D（ms）	例程 E（ms）
a.	2	20	80	10	70	5
b.	16	4	14	2	12	2

1.16.1 [10] <1.8> 求总的执行时间。如果例程 A，C 和 E 的时间改进 15%，总的执行时间能减少多少？

1.16.2 [10] <1.8> 如果例程 B 的时间改进 10%，总的执行时间能减少多少？

1.16.3 [10] <1.8> 如果例程 D 的时间改进 10%，总的执行时间能减少多少？

1.16.1

处理器数为 2 时，$T_a = 185ms$，处理器数为 16 时 $T_b = 34ms$

处理器数为 2 时，$\Delta T_{a1} = 0.15 \times (20+10+5)ms = 5.25ms$

处理器数为 16 时，$\Delta T_{b1} = 0.15 \times (4+2+2)ms = 1.2ms$

1.16.2

处理器数为 2 时，$\Delta T_{a2} = 0.1 \times 80ms = 8ms$

处理器数为 16 时，$\Delta T_{b2} = 0.1 \times 14ms = 1.4ms$

1.16.3

处理器数为 2 时，$\Delta T_{a3} = 0.1 \times 70ms = 7ms$

处理器数为 16 时，$\Delta T_{b3} = 0.1 \times 12ms = 1.2ms$

多处理器系统中的执行时间可分成例程计算时间加处理器之间的通信时间。下表给出了例程计算时间和通信时间。在这种情况下，通信时间是总时间的重要组成部分。

处理器数	例程 A（ms）	例程 B（ms）	例程 C（ms）	例程 D（ms）	例程 E（ms）	通信（ms）
2	20	78	9	65	4	11
4	12	44	4	34	2	13
8	1	23	3	19	3	17
16	4	13	1	10	2	22
32	2	5	1	5	1	23
64	1	3	0.5	1	1	26

1.16.4 [10] <1.8> 每当处理器数量加倍时，求新的计算时间与旧的计算时间之比和新与旧的通信时间之比。

1.16.5 [5] <1.8> 用比值的几何平均值，推算在 128 台处理器的系统中的计算时间和通信时间。

1.16.6 [10] <1.8> 求在 1 个处理器系统中的计算时间和通信时间。

处理器数	计算时间	通信时间	计算时间比	通信时间比
2	176	11	–	–
4	96	13	0.5455	1.1818
8	49	17	0.5104	1.3077
16	30	22	0.6122	1.2941
32	14	23	0.4667	1.0455
64	6.5	26	0.4643	1.1304
128	3.36	30.88	0.517	1.1877
1	340.43	0	–	–

2.8 [10] <2.2, 2.3> 将以下 RISC-V 指令转换成 C 代码。假设变量 f、g、h、i 和 j 分别分配给寄存器 x5、x6、x7、x28 和 x29。假设数组 A 和 B 的基地址分别在寄存器 x10 和 x11 中。

```
addi  x30, x10, 8
addi  x31, x10, 0
sd    x31, 0(x30)
ld    x30, 0(x30)
add   x5,  x30, x31
```

2.9 [20] <2.2, 2.5> 对于练习 2.8 中的每条 RISC-V 指令，写出操作码（op）、源寄存器（rs1）和目标寄存器（rd）字段的值。对于 I 型指令，写出立即数字段的值，对于 R 型指令，写出第二个源寄存器（rs2）的值。对于非 U 型和 UJ 型指令，写出 funct3 字段，对于 R 型和 S 型指令，写出 funct7 字段。

	opcode	rs1	rd	Type,Imm/rs2	funct3	funct7
addi	0010011	01010	11110	I,8	000	
addi	0010011	01010	11111	I,0	000	
sd	0100011	11110		S	011	
ld	0000011	11110	11110	I,0	011	
add	0110011	11110	00101	R,11111	000	0000000

2.24

2.24 考虑以下 RISC-V 循环：

```
LOOP:  beq  x6, x0, DONE
       addi x6, x6, -1
       addi x5, x5, 2
       jal  x0, LOOP
DONE:
```

2.24.1 [5] <2.7> 假设寄存器 x6 初始化为 10。寄存器 x5 的最终值是多少（假设 x5 初始值为零）？

2.24.2 [5] <2.7> 对于上面的循环，编写等效的 C 代码。假设寄存器 x5 和 x6 分别是整型 acc 和 i。

2.24.3 [5] <2.7> 对于上面用 RISC-V 汇编语言编写的循环，假设寄存器 x6 初始化为 N。总共执行了多少条 RISC-V 指令？

2.24.4 [5] <2.7> 对于上面用 RISC-V 汇编语言编写的循环，将指令“beq x6, x0, DONE”替换为“blt x6, x0, DONE”指令并写出等效的 C 代码。

2.24.1
20，10次循环，每次加2

2.24.2
```
acc = 0;
for(i=10; i!=0; i--){
    acc += 2;
}
```

2.24.3
4N+1，每次循环执行4条指令，共N次循环

2.24.4
```
acc = 0;
for(i=10; i>=0; i--){
    acc += 2;
}
```

2.35 考虑以下代码：

```
lb x6, 0(x7)
sd x6, 8(x7)
```

假设寄存器 x7 包含地址 0x10000000，且地址中的数据是 0x1122334455667788。

2.35.1 [5] <2.3, 2.9> 在大端对齐的机器上 0×10000008 中存储的是什么值？

2.35.2 [5] <2.3, 2.9> 在小端对齐的机器上 0×10000008 中存储的是什么值？

小端序，低位低地址，高位高地址；大端序，低位高地址，高位低地址
以**字节**为单位编址，某个多字节数据的地址是其最低字节的地址

lb x6, 0(x7)　// lb取位于0x10000000处的字节，并进行符号扩展至64位，存入x6
sd x6, 8(x7)　// sd将x6中的8个字节存入0x10000008~ 0x1000000F处的8个内存单元

	00h	01h	02h	03h	04h	05h	06h	07h
大端序	11h	22h	33h	44h	55h	66h	77h	88h
小端序	88h	77h	66h	55h	44h	33h	22h	11h

	lb	x6	08h	09h	……	0Fh
大端序	11h	0000000000000011h	00h	00h	0000000000h	11h
小端序	88h	FFFFFFFFFFFFFF88h	88h	FFh	FFFFFFFFFFh	FFh

2.40 假设对于一个给定程序，70% 的执行指令是算术指令，10% 是加载 / 存储指令，20% 是分支指令。

2.40.1 [5] <1.6, 2.13> 假设算术指令需要 2 周期，加载 / 存储指令需要 6 周期，而一条分支指令需要 3 周期，求平均 CPI。

2.40.2 [5] <1.6, 2.13> 对于性能提高 25%，如果加载 / 存储和分支指令都没有改进，一条算术指令平均需要多少周期？

2.40.3 [5] <1.6, 2.13> 对于性能提高 50%，如果加载 / 存储和分支指令都没有改进，一条算术指令平均需要多少周期？

2.40.1

$CPI = 0.7 * 2 + 0.1 * 6 + 0.2 * 3 = 2.6$

2.40.2

$performance = 1/T$

$P\_new = 1.25P,\ T\_new = T/1.25 = 0.8T$

$0.7x + 1.2 = 2.6 * 0.8,\ x = 1.257$

2.40.3

同2.40.2

x=0.7619