指令序列构造实验实验报告

组员：仇成宇 曾宪伟 赵泽锴 邹旻昊

关于实验报告中的所有实验有两点需要声明

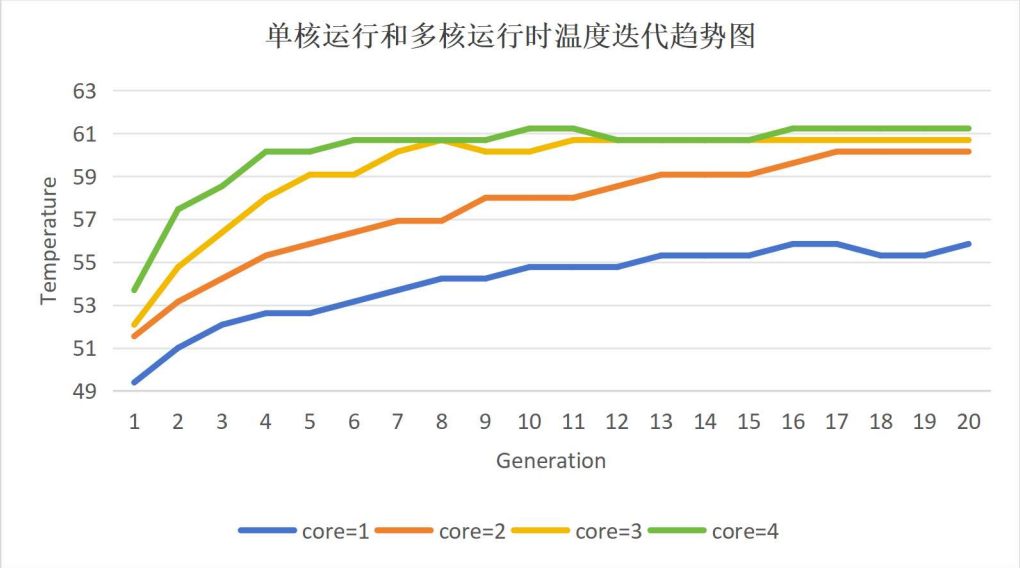
（1）由于无法在两次实验之间等待过长的时间，不能保证每次实验初始温度完全一致。但是我们保证每次实验开始前的初始温度在 49 度以下和 47 度以上，以保证实验效果。

（2）由于不同实验在不同时间完成，实验存在很多非确定性因素，即使参数相同也不排除结果不同的可能（甚至可能差距很大，我们只能保证所给出的数据在我们进行实验的时候发生，不保证能够精确复现）。

一、 实验 1

本实验测量遗传算法单核运行和多核运行时的处理器温度情况并作图。从表格及图中可以看出，最高温度随绑定核数增加而增加，因此在之后的实验中均绑定 4 核以获得最高温度。

	1 个核	2 个核	3 个核	4 个核
温度	55.844	60.148	60.686	61.224

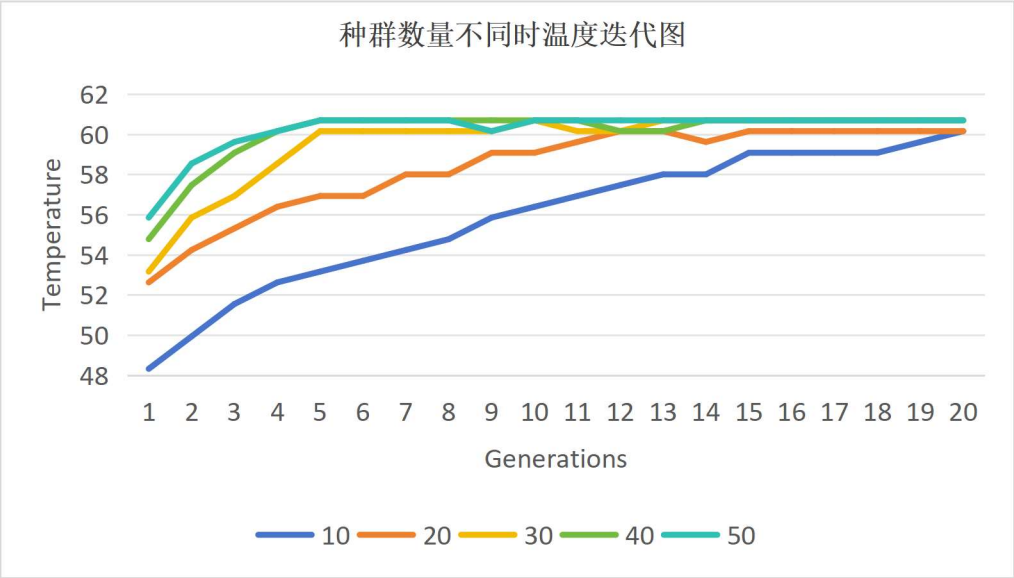


二、 实验 2

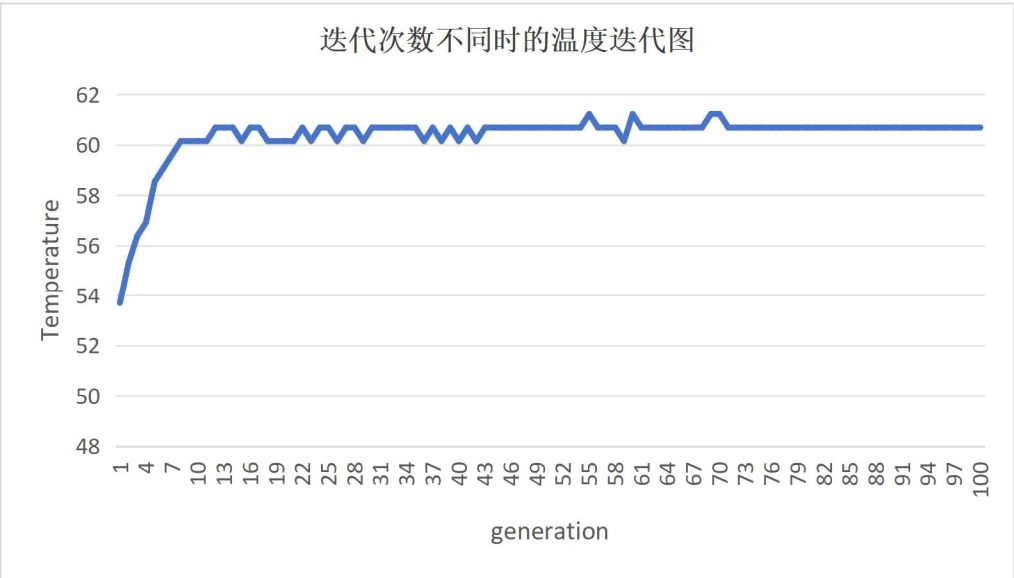
为保证本实验在后续实验中的价值，本实验中的指令集采用的是实验 3 中经过扩展后的指令集，相应的也更改了采用的处理器架构，我们相信这样测量出的结果会对于后续扩展指令后的实验更具有参考意义。

本实验测量遗传算法在不同的参数下，四个核同时运行指令序列片段时的处理器温度情况并作图。

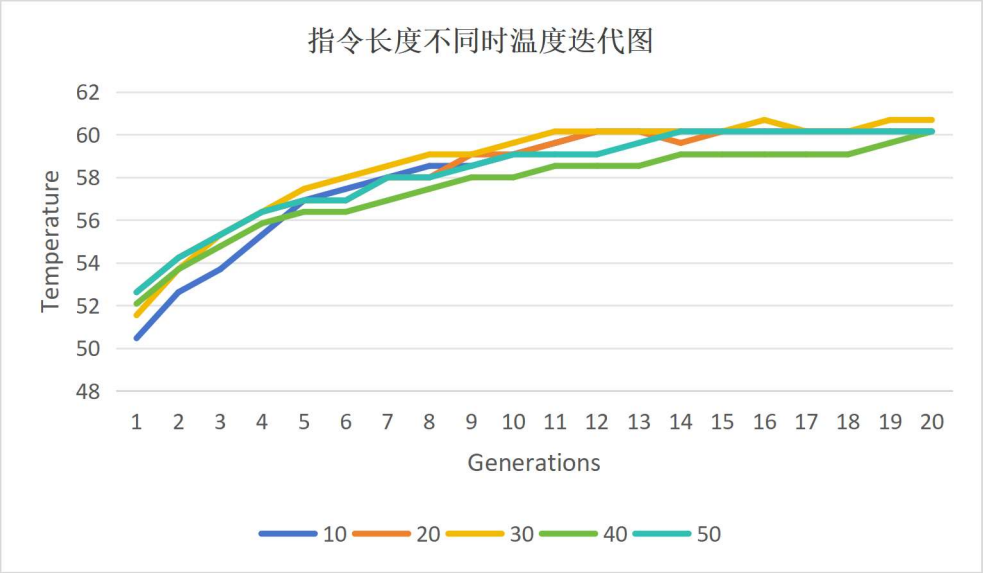
每代种群数量	温度
10	60.148
20	60.148
30	60.686
40	60.686
50	60.686



迭代次数	温度
20	60.686
40	60.686
60	61.224
80	61.224
100	61.224



指令序列长度	温度
10	60.148
20	60.148
30	60.686
40	60.148
50	60.148

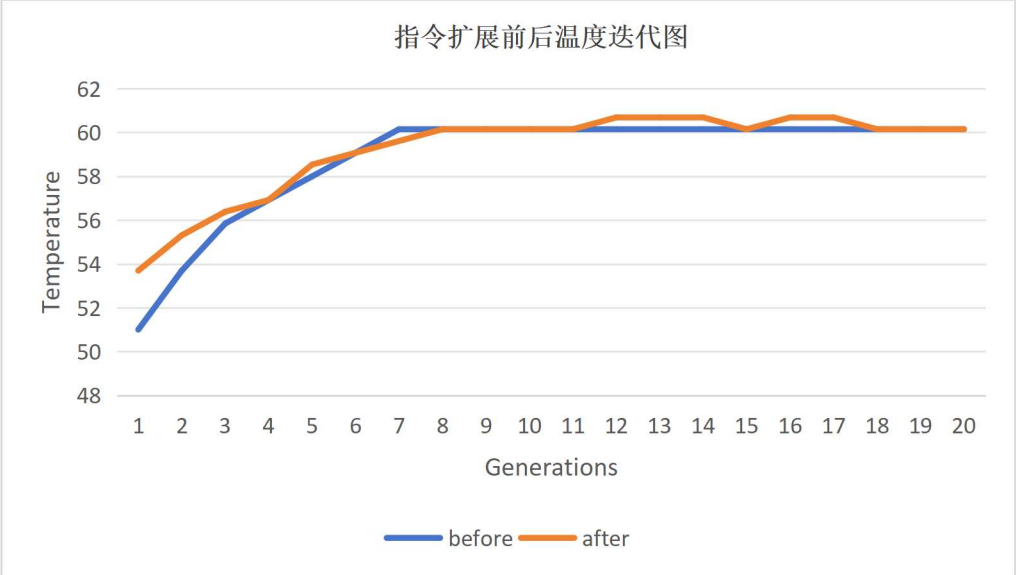


三、 实验 3

本实验扩展指令集测试，观察温度变化并作图。由于本实验目的在于提高处理器温度，因此想到用更加复杂的浮点数和 simd 指令进行测试。

首先我们将架构由 armv6 更改为 armv8-a，并且设置 fpu 为 neon 支持。之后我们在指令集中添加了 usub8，vmla.f32，vmla.f64，vmov.f64，vcvt.f64.s32，vcvt.f32.f64，vstr.64，vldr.64，vldr.32，vstr.32,vdiv.f32,vdiv.f64,vmul.f32,vmul.f64,vadd.f32,vadd.f64。由于树莓派不支持大部分 simd-advanced 指令，所以能够加入的指令有限。实验显示在扩展指令集后刚开始的温度提升更快，且整体来看最高温度更高。

	温度
扩展指令集前	60.012
扩展指令集后	60.686



四、 实验 4

我们首先根据数据和整理的结果，总结从实验 1 到实验 3 对于指令序列构造的重要结论：

Conclusion：

1、运行指令的总数是影响温度升高快慢的最重要的因素，在遗传算法中运行指令总数和轮数、种群大小成正比。在各个参数中指令长度对温度升高的影响最小。

2、按照目前每次运行 0.2 秒就杀掉的策略，温度在一段时间的迭代后达到稳定状态并开始波动，之后可能会小幅度增加。稳定状态的温度与遗传算法的各个参数关系不大，而且扩展指令集后并没有很大提升。

据此我们对遗传算法加以分析：按照遗传算法的策略，整个选择过程会偏向于使得指令序列在运行后 CPU 温度最高的程序加以进化。这里的局限性在于：（1）本指令序列运行前的 CPU 温度是不经过考虑的，不同指令序列运行前的 CPU 温度可能不同（2）本指令序列运行后的 CPU 温度高并不代表这本指令序列不断运行的温度累积能力强，这还要考虑 CPU 本身的调控和适应。

针对第一点，由于温度增加本身就和当前温度有很大的关系（温度越高增加越慢），因此如果要更改遗传算法需要设计较复杂的温度模型，这里我们倾向于认为遗传算法整体还是会选择耗能较高的指令，只是不能默认温度最高的指令序列就是最好的指令序列。因此我们采取的策略是，当温度稳定后，手动分析每次迭代的指令序列，进而挑选较好的，并且大规模进行温度累积的测试（即 exp_test），从中取优。

针对第二点，我们尝试对每条指令构造完全由该条指令构成的指令序列，要求不存在流水线卡顿和数据依赖，并且在运行固定步数之后测量温度。该结果能够更好的反映每条指令的温度累积能力，从而便于在手动调整指令时大概明白指令的功耗。下表中列出了一部分的测量结果。

指令	温度
Vadd.f64	60.686
Vmul.f64	62.3
Vdiv.f64	60.148

Vldr.64	60.686
Vstr.32	60.686
Vdiv.f32	59.61
Vmul.f32	60.686
Vadd.f32	61.224
Mov reg1 reg2	63.376
Mov reg1 num0T255	61.762
ldr	61.224
str	60.686

基于上面的思考和测试，我们拟定了针对本实验的指令序列构造策略：首先，在指令长度尽可能短（定为 10，使得遗传算法更容易获得最优解）的情况下，进行大规模的遗传迭代（100 轮每次，多次迭代），然后在稳定期的生成汇编代码中进行手动选择和手动调优（包括将简单指令替换为同类型的更复杂指令，并去除数据依赖），之后对生成的这一系列指令序列进行温度测试。除上述最终方案外，这里我们还尝试了遗传算法参数的调整以及手动从头开始构造指令序列，同时也尝试了多个指令长度如 12、13、15，以及缩减指令集，但是效果不佳。

下面直接给出最终我们构造的代码并进行分析：

```
and     r7 ,r6 ,r12
usub8   r5 ,r10 ,r0
str     r4 ,[r13]
vcvt.f32.f64    s2 ,d17
add     r12 ,r11 ,r12
mov     r9 ,r1
add     r0 ,r5 ,r3
vmla.f64        d9 ,d5 ,d5
usub8   r6 ,r10 ,r10
vcvt.f64.s32    d7 ,s6
```

这份汇编代码是在遗传算法迭代的基础经过手动优化得到的，其优势有：（1）没有数据依赖，不会造成流水线卡顿（2）使用了较为复杂的 usub8 和 vmla 指令（3）使用了上面测试中累计温度能力较强的 mov。其温度结果如下表所示：

Example4.s 运行温度	我们构造的指令序列运行温度
60.713	62.838

需要注意的是，并不是满足上述提到的优点的指令序列就可以达到高温度，这其中有很多因素我们目前无法知晓。例如，我们试图将这里面的指令顺序进行替换，或者仅仅增加一条指令，发现会对温度造成较大的影响，具体原因未知。