作业一：编译器优化选项

唐亚周 519021910804

提交要求：作业文档使用PDF格式提交。如果问题2使用开源软件，则在文档中注明软件仓库地址或下载链接，仅提交单独的PDF文档即可；如果问题2自行编写代码，则将PDF文档和源代码打包成一个ZIP/RAR压缩文件提交。

1. 针对GCC或Clang编译器缺省的优化选项（-O0、-O1、-O2、-O3、-Os、-Og、-Ofast、-Oz等），采用一个表格，枚举出其实际实施的优化策略（即optimization flag）。如果一个编译选项包含某一优化策略，则在相应的单元格内打勾（√）；若不包含，不需要打叉。

   答：以下为 GCC 的优化选项：

优化策略	-O0	-O1	-O2	-O3	-Os	-Og	-Ofast	-Oz
-fauto-inc-dec		√	√	√	√	√	√	√
-fbranch-count-reg		√	√	√	√		√	√
-fcombine-stack-adjustments		√	√	√	√	√	√	√
-fcompare-elim		√	√	√	√	√	√	√
-fcprop-registers		√	√	√	√	√	√	√
-fdce		√	√	√	√	√	√	√
-fdefer-pop		√	√	√	√	√	√	√
-fdelayed-branch		√	√	√	√		√	√
-fdse		√	√	√	√		√	√
-fforward-propagate		√	√	√	√	√	√	√
-fguess-branch-probability		√	√	√	√	√	√	√
-fif-conversion		√	√	√	√		√	√
-fif-conversion2		√	√	√	√		√	√
-finline-functions-called-once		√	√	√	√		√	√
-fipa-modref		√	√	√	√	√	√	√
-fipa-profile		√	√	√	√	√	√	√
-fipa-pure-const		√	√	√	√	√	√	√
-fipa-reference		√	√	√	√	√	√	√
-fipa-reference-addressable		√	√	√	√	√	√	√
-fmerge-constants		√	√	√	√	√	√	√
-fmove-loop-invariants		√	√	√	√		√	√
-fmove-loop-stores		√	√	√	√		√	√
-fomit-frame-pointer		√	√	√	√	√	√	√
-freorder-blocks		√	√	√	√	√	√	√
-fshrink-wrap		√	√	√	√	√	√	√
-fshrink-wrap-separate		√	√	√	√	√	√	√
-fsplit-wide-types		√	√	√	√	√	√	√
-fssa-backprop		√	√	√	√	√	√	√
-fssa-phiopt		√	√	√	√		√	√

优化策略	-O0	-O1	-O2	-O3	-Os	-Og	-Ofast	-Oz
-ftree-bit-ccp		√	√	√	√		√	√
-ftree-ccp		√	√	√	√	√	√	√
-ftree-ch		√	√	√	√	√	√	√
-ftree-coalesce-vars		√	√	√	√	√	√	√
-ftree-copy-prop		√	√	√	√	√	√	√
-ftree-dce		√	√	√	√	√	√	√
-ftree-dominator-opts		√	√	√	√	√	√	√
-ftree-dse		√	√	√	√		√	√
-ftree-forwprop		√	√	√	√	√	√	√
-ftree-fre		√	√	√	√	√	√	√
-ftree-phiprop		√	√	√	√	√	√	√
-ftree-pta		√	√	√	√		√	√
-ftree-scev-cprop		√	√	√	√	√	√	√
-ftree-sink		√	√	√	√	√	√	√
-ftree-slsr		√	√	√	√	√	√	√
-ftree-sra		√	√	√	√		√	√
-ftree-ter		√	√	√	√	√	√	√
-funit-at-a-time		√	√	√	√	√	√	√
-falign-functions			√	√			√	
-falign-jumps			√	√			√	
-falign-labels			√	√			√	
-falign-loops			√	√			√	
-fcaller-saves			√	√	√		√	√
-fcode-hoisting			√	√	√		√	√
-fcrossjumping			√	√	√		√	√
-fcse-follow-jumps			√	√	√		√	√
-fcse-skip-blocks			√	√	√		√	√
-fdelete-null-pointer-checks			√	√	√		√	√
-fdevirtualize			√	√	√		√	√
-fdevirtualize-speculatively			√	√	√		√	√
-fexpensive-optimizations			√	√	√		√	√
-ffinite-loops			√	√	√		√	√
-fgcse			√	√	√		√	√
-fgcse-lm			√	√	√		√	√
-fhoist-adjacent-loads			√	√	√		√	√
-finline-functions			√	√	√		√	√
-finline-small-functions			√	√	√		√	√
-findirect-inlining			√	√	√		√	√
-fipa-bit-cp			√	√	√		√	√

优化策略	-O0	-O1	-O2	-O3	-Os	-Og	-Ofast	-Oz
-fipa-cp			√	√	√		√	√
-fipa-icf			√	√	√		√	√
-fipa-ra			√	√	√		√	√
-fipa-sra			√	√	√		√	√
-fipa-vrp			√	√	√		√	√
-fisolate-erroneous-paths-dereference			√	√	√		√	√
-flra-remat			√	√	√		√	√
-foptimize-sibling-calls			√	√	√		√	√
-foptimize-strlen			√	√	√		√	√
-fpartial-inlining			√	√	√		√	√
-fpeephole2			√	√	√		√	√
-freorder-blocks-algorithm=stc			√	√			√	
-freorder-blocks-and-partition			√	√	√		√	√
-freorder-functions			√	√	√		√	√
-frerun-cse-after-loop			√	√	√		√	√
-fschedule-insns			√	√	√		√	√
-fschedule-insns2			√	√	√		√	√
-fsched-interblock			√	√	√		√	√
-fsched-spec			√	√	√		√	√
-fstore-merging			√	√	√		√	√
-fstrict-aliasing			√	√	√		√	√
-fthread-jumps			√	√	√		√	√
-ftree-builtin-call-dce			√	√	√		√	√
-ftree-loop-vectorize			√	√	√		√	√
-ftree-pre			√	√	√		√	√
-ftree-slp-vectorize			√	√	√		√	√
-ftree-switch-conversion			√	√	√		√	√
-ftree-tail-merge			√	√	√		√	√
-ftree-vrp			√	√	√		√	√
-fvect-cost-model=very-cheap			√		√			√
-fgcse-after-reload				√			√	
-fipa-cp-clone				√			√	
-floop-interchange				√			√	
-floop-unroll-and-jam				√			√	
-fpeel-loops				√			√	
-fpredictive-commoning				√			√	
-fsplit-loops				√			√	
-fsplit-paths				√			√	
-ftree-loop-distribution				√			√	

优化策略	-O0	-O1	-O2	-O3	-Os	-Og	-Ofast	-Oz
-ftree-partial-pre				√			√	
-funswitch-loops				√			√	
-fvect-cost-model=dynamic				√			√	
-fversion-loops-for-strides				√			√	
-finline-functions					√			√
-ffast-math							√	
-fallow-store-data-races							√	
-fstack-arrays							√	

参考：

(a) Optimize Options (Using the GNU Compiler Collection (GCC))
(b) gcc(1) [osx man page]

注意：

(a) 最后3个 `-Ofast` 选项独有的优化策略在 `-fmax-stack-var-size` 和 `-fno-protect-parens` 选项被指定打开时，会被关闭。

(b) `-Oz` 与 `-Os` 在优化策略上的具体区别，官网上并没有详细说明。但 `-Oz` 选项是APPLE公司的操作系统中特有的，且比 `-Os` 选项在体积优化上更为激进（牺牲性能）。比如，`-Oz` 选项开启后，编码成更少字节的指令比执行周期更少的长指令更受欢迎。

> **-Oz** (APPLE ONLY) Optimize for size, regardless of performance. **-Oz** enables the same optimization flags that **-Os** uses, but **-Oz** also enables other optimizations intended solely to reduce code size. In particular, instructions that encode into fewer bytes are preferred over longer instructions that execute in fewer cycles. **-Oz** on Darwin is very similar to **-Os** in FSF distributions of GCC. **-Oz** employs the same inlining limits and avoids string instructions just like **-Os**.

2. 准备一个样例程序， 使用问题1中考察的**所有**编译选项对目标程序进行编译，描述最终编译后的**二进制程序文件大小**，及程序**运行性能**上的差异。可以自行编写代码，也可以选择开源软件进行评测（如：https://github.com/arbenson/fast-matmul）。如果自行编写代码，建议代码不要过于简单，否则不同优化选项的编译结果可能差异不大。

   答：利用上课所学的7种优化情况的示例代码编写测试代码（`hw1p2.cpp`），然后编写Makefile采用不同的编译优化选项进行编译，再编写测试脚本（`test.sh`）进行测试。测试结果与分析如下：

   （由于我的操作系统不是MacOS，因此没有测试 `-Oz` 优化选项。）

   (a) **运行性能：**

```
 /mnt/D/SJ/3-Junior-2/SE3/P/assignment1 ./test.sh
-O0
The program's output is 1024071094
The program took 89.0979 ms.
-O1
The program's output is 1024071094
The program took 29.4534 ms.
-O2
The program's output is 1024071094
The program took 27.4475 ms.
-O3
The program's output is 1024071094
The program took 19.3417 ms.
-Ofast
The program's output is 1024071094
The program took 18.9318 ms.
-Og
The program's output is 1024071094
The program took 31.6677 ms.
-Os
The program's output is 1024071094
The program took 26.1367 ms.
```

可以看到，`-O0`、`-O1`、`-O2`、`-O3`、`-Ofast` 这几个优化选项编译出的程序，其运行性能大致呈递增趋势。而 `-Og` 选项开启后，编译出的程序的性能与 `-O1` 相近，`-Os` 选项开启后，编译出的程序的性能与 `-O2` 相近。

(b) **文件大小：**

```
 /mnt/D/SJ/3-Junior-2/SE3/P/assignment1 ls -l | grep "hw1p2_O"
-rwxrwxrwx 1 adswt518 root 18584  2月  27 20:28 hw1p2_O0
-rwxrwxrwx 1 adswt518 root 16976  2月  27 20:28 hw1p2_O1
-rwxrwxrwx 1 adswt518 root 17192  2月  27 20:28 hw1p2_O2
-rwxrwxrwx 1 adswt518 root 17160  2月  27 20:28 hw1p2_O3
-rwxrwxrwx 1 adswt518 root 18728  2月  27 20:28 hw1p2_Ofast
-rwxrwxrwx 1 adswt518 root 17048  2月  27 20:28 hw1p2_Og
-rwxrwxrwx 1 adswt518 root 16976  2月  27 20:28 hw1p2_Os
```

可以看到，`-Os` 和 `-O1` 选项所编译出的程序最小。其它情况中，`-O2`、`-O3`、`-Ofast` 这几个选项开启后，为了性能而牺牲了代码的精简；`-Og` 选项为了调试的方便而牺牲了代码的精简；`-O0` 选项则因为完全没有优化，部分死代码没有消除，导致编译出的文件较大。