



南開大學
Nankai University

南 开 大 学

计 算 机 学 院

预习报告

丁屹

年级：2020 级

专业：计算机科学与技术

2022 年 9 月 19 日

摘要

关键字：XX、YY、ZZ

目录

一、 作业 1	1
二、 作业 2	3
三、 作业 3	4
四、 作业 4	5

一、 作业 1

对于以下两种代码，选择 LOOP #2 可以得到更快的执行速度。

```
1  /* LOOP #1 */
2  int i;
3  for (i = 0; i < N; ++i) {
4      a[i] *= 2000;
5      a[i] /= 10000;
6  }
```

```
1  /* LOOP #2 */
2  int* b = a;
3  int i;
4  for (i = 0; i < N; ++i) {
5      *b *= 2000;
6      *b /= 10000;
7      ++b;
8  }
```

在 x86_64 CPU, Arch Linux 系统使用 GCC 12.2.0 编译器测试，其中 N 取值 1000000000，取 4 次用时的平均值，计时器使用 `clock_gettime(CLOCK_MONOTONIC, &ts)`，代码位于 `src/loop1.c` 与 `src/loop2.c`。

如果不打开 -O 优化，可以得到 LOOP #1 平均用时 5049361472 ns，得到 LOOP #2 平均用时 2849261662 ns。LOOP #2 速度更快。

```
1  a:
2      .zero    4000000000
3  main:
4      push    rbp
5      mov     rbp, rsp
6      mov     DWORD PTR [rbp-4], 0
7      jmp     .L2
8  .L3:
9      mov     eax, DWORD PTR [rbp-4]
10     cdqe
11     mov     eax, DWORD PTR a[0+rax*4]
12     imul    edx, eax, 2000
13     mov     eax, DWORD PTR [rbp-4]
14     cdqe
15     mov     DWORD PTR a[0+rax*4], edx
16     mov     eax, DWORD PTR [rbp-4]
17     cdqe
```

```

18     mov     eax, DWORD PTR a[0+rax*4]
19     movsx   rdx, eax
20     imul    rdx, rdx, 1759218605
21     shr     rdx, 32
22     sar     edx, 12
23     sar     eax, 31
24     sub     edx, eax
25     mov     eax, DWORD PTR [rbp-4]
26     cdqe
27     mov     DWORD PTR a[0+rax*4], edx
28     add     DWORD PTR [rbp-4], 1
29     .L2:
30     cmp     DWORD PTR [rbp-4], 999999999
31     jle     .L3
32     mov     eax, 0
33     pop     rbp
34     ret

1     a:
2     .zero   4000000000
3     main:
4     push    rbp
5     mov     rbp, rsp
6     mov     QWORD PTR [rbp-8], OFFSET FLAT:a
7     mov     DWORD PTR [rbp-12], 0
8     jmp     .L2
9     .L3:
10    mov     rax, QWORD PTR [rbp-8]
11    mov     eax, DWORD PTR [rax]
12    imul    edx, eax, 2000
13    mov     rax, QWORD PTR [rbp-8]
14    mov     DWORD PTR [rax], edx
15    mov     rax, QWORD PTR [rbp-8]
16    mov     eax, DWORD PTR [rax]
17    movsx   rdx, eax
18    imul    rdx, rdx, 1759218605
19    shr     rdx, 32
20    sar     edx, 12
21    sar     eax, 31
22    sub     edx, eax
23    mov     rax, QWORD PTR [rbp-8]
24    mov     DWORD PTR [rax], edx
25    add     QWORD PTR [rbp-8], 4
26    add     DWORD PTR [rbp-12], 1

```

```
27  .L2:
28      cmp     DWORD PTR [rbp-12], 999999999
29      jle     .L3
30      mov     eax, 0
31      pop     rbp
32      ret
```

比较二者的汇编代码可以发现主要的区别在于：LOOP #1 在每次访问数组时都会计算 $0+rax*4$ ，做下标转换；而 LOOP #2 中每次只对指针 $[rbp-8]+4$ ，计算量更小。

如果开启-O2 优化，可以得到 LOOP #1 平均用时 1652242168 ns，得到 LOOP #2 平均用时 1664580829 ns。可以认为 LOOP #1 与 LOOP #2 没有性能差距。此时二者汇编代码没有区别，优化为使用 SIMD 超标量技术加速。

二、 作业 2

分词

- Model (field)
- = (boolean operator)
- " (string begin)
- Civic (string content)
- " (string end)
- AND (boolean operator)
- Year (field)
- = (boolean operator)
- " (string begin)
- 2001 (string content)
- " (string end)

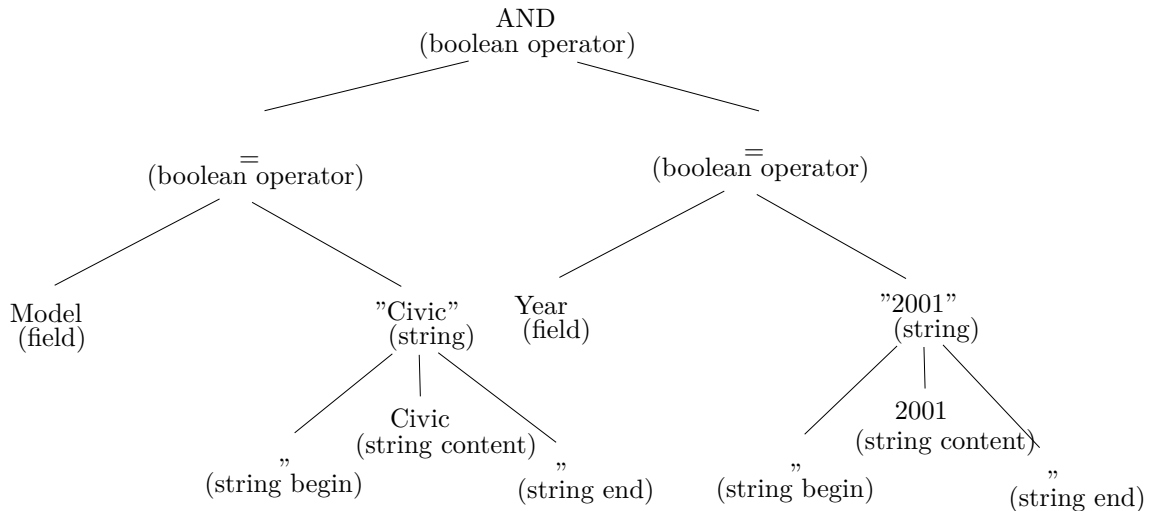


图 1: 语法分析树

三、 作业 3

使用 splint 可以得到如下输出

```

1 Splint 3.1.2a --- May 25 2020
2
3 src/static-check.c: (in function f)
4 src/static-check.c:13:10: Stack-allocated storage &loc reachable from return
5                             value: &loc
6   A stack reference is pointed to by an external reference when the function
7   returns. The stack-allocated storage is destroyed after the call, leaving a
8   dangling reference. (Use -stackref to inhibit warning)
9 src/static-check.c:13:10: Immediate address &loc returned as implicitly only:
10                             &loc
11   An immediate address (result of & operator) is transferred inconsistently.
12   (Use -immediatetrans to inhibit warning)
13 src/static-check.c:13:15: Stack-allocated storage *x reachable from parameter x
14   src/static-check.c:12:3: Storage *x becomes stack-allocated storage
15 src/static-check.c:13:15: Function returns with global glob referencing
16                             released storage
17   A global variable does not satisfy its annotations when control is
18   transferred. (Use -globstate to inhibit warning)
19   src/static-check.c:13:10: Storage glob released
20 src/static-check.c: (in function h)
21 src/static-check.c:18:7: Comparison of unsigned value involving zero: i >= 0
22   An unsigned value is used in a comparison with zero in a way that is either a
23   bug or confusing. (Use -unsignedcompare to inhibit warning)
24 src/static-check.c:18:7: Variable i used before definition
25   An rvalue is used that may not be initialized to a value on some execution
  
```

```
26 path. (Use -usedef to inhibit warning)
27 src/static-check.c:7:6: Variable exported but not used outside static-check:
28     glob
29 A declaration is exported, but not used outside this module. Declaration can
30 use static qualifier. (Use -exportlocal to inhibit warning)
31
32 Finished checking --- 7 code warnings
```

- 在函数 `f` 内, 返回了指向局部变量 `loc` 的指针, 可能会导致释放后使用
- 在函数 `f` 内, 传入的参数 `x` 指向的指针 `*x` 被写入了局部变量 `loc` 的地址, 函数返回后访问 `**x` 会导致释放后使用
- 在函数 `h` 内, 未初始化变量 `i` 就访问其值
- 在函数 `h` 内, 变量 `i` 类型是无符号整数, 只能走 $i \geq 0$ 分支, 另一分支无意义
- 在函数 `firstChar1` 内, 没有对传入指针做检查, 可能会导致解引用 `NULL` (此处 `splint` 不认为有问题)

四、 作业 4