编译原理实验报告

——编译器调研

实验编号 一

实验名称 编译器调研

实验室名称 软件学院实验室

班级 软件学院 14 级二班

学号-姓名 刘嘉洋-1412620

实验日期 2016 年 9 月 1 日

至 2016 年 10 月 10 日

评分教师_____ 实验报告成绩_____

评分日期_____年_____月_____日

一、实验目的：

- 调研 C 编译器语言特性及功能
- 定义自制编译器的 C 语言子集
- 思考如何将 C 语言子集特性转换为汇编程序

二、调研对象

GCC 6.2 (August 22,2016)

三、实验要求

- 调研你最熟悉、最常使用的 C 编译器：它支持哪些主要的 C（C++）语言特性，特别是支持哪些最新语言发展的新特性？它具有哪些功能，特别是吻合计算机领域新发展的一些新功能（对 CPU 新架构的支持、调试/优化代码的生成、自动并行化、跨平台等等）？
- 定义你的编译器支持的 C 语言子集——预习教材第 2 章及第 2 章讲义中的 2.2 节，用上下文无关文法描述你的 C 语言子集。
- 思考：结合预备工作 1，你的 C 语言子集中的每个特性（语句）如何转换为汇编程序？注意：以赋值语句为例，一个编译器要做的不是将特定的 i=1; 转换为汇编程序，而是要能将所有可能的赋值语句转换为汇编程序。

四、编译器调研

（我在这一个月的实验过程中，阅读了GCC 6.2 changes原版文档，通过翻译和探索，虽然仍有大部分由于知识水平的尚不成熟而不能理解，但很大程度上拓宽了我的见识面，更让我有了对C语言族很多新的认知。）

Caveats警告

- C++的默认模式已改为-std=gnu++14，而不是-std=gnu++98。
- 旧的系统SH5 / SH64 (sh64-*-*)在下一个版本中将不再支持。
- AVR单片机端口需要binutils version 2.26.1以上的版本。

.....

General Optimizer Improvements常用优化器的改进

- UndefinedBehaviorSanitizer有了新的处理选项，-fsanitize=bounds-strict, 支持了数组边界的强检查。 特别的是，同时也支持-fsanitize=bounds。
- 基于类型的别名分析器消除了通向不同指针的模糊含义。这提升了高级C++程序的别名精确性20-30%。
- 别名分析现在准确支持weakref和alias属性。这将使得在一个翻译单元中访问变量和它的别名成为可能。
- C++成员函数中的this指针现在默认为非空，这消除了常见的空指针判断以及一些非一致性的编写方式(例如Qt-5, Chromium, KDevelop)。可使用-fno-delete-null-pointer-checks作为临时措施。错误的代码段可通过使用-fsanitize=undefined来找到。
- 链接时优化(Link-time optimization)的提升:
  - 类型合并被改进以处理C和Fortran的互操作性规则。除了一个特例：CHARACTER(KIND=C_CHAR)和char 不可互操作，因为其为数组而char是标量，可用INTEGER(KIND=C_SIGNED_CHAR)替代。
  - 更多的类型信息在链接时被保留，通过链接时优化提升了类型别名分析的精确性。
  - LTO对象文件的大小被减小了约11% (通过Firefox 46.0编译测量)。
  - 链接时并行化(使用-flto=n)通过减小分区程序的数据流被显著提升。IL流的大小减少到66%(通过Firefox 46.0编译测量)。
  - 链接器插件被扩展来支持传递二进制类型信息到GCC后台的(也可以通过-flinker-output手动控制)。这使得正确的安装代码生成器成为可能，并支持增量链接（编译器为提升链接速度而增加的功能，为目标的（函数）代码“预留一部分空间”，当代码修改后，只需要修改这一部分对象代码即可快速完成编译与链接）。通过gcc -r 的LTO对象的增量链接现在支持。先有两种方式执行增量链接:

    1 通过ld －r：生成自动将所有对象文件合并起来的一个对象文件。这样延迟了实际的到最终链接的链接优化，并以此允许了整个程序的优化。然而链接带有这样对象文件的二进制程序将会更慢。

    2 通过gcc －r： will将使得链接时优化和产生最终的二进制代码到对象文件。链接这样的对象文件更快但是失去的整个程序优化的好处。
- 程见优化(Inter-procedural optimization)的提升:
  - 基础的跳转线程现在在轮廓建造和内联分析前执行。
  - 功能克隆法(Function cloning)消除了没有被使用的函数参数。

.....

New Languages and Language specific improvements新的语言以及语言的优化

相比GCC 5, GCC 6发布系列包含了显著提升的OpenACC 2.0a specification实现。亮点包括:

- 除了执行单线程的主机反馈(host-fallback), 卸下(offloading)现在对x86_64和PowerPC 64-bit little-endian GNU/Linux主机系统上的nvptx (Nvidia GPUs)提供支持。对于nvptx卸下,包括OpenACC并行构建, 执行模块允许任意数群至多32个worker和32个vector。
- 开始支持OpenACC核心构建的并行化执行:
  - 内核区的并行化可通过-fopenacc与-O2或更高来开启。
  - 代码卸下到多个群，但只通过长度为1的一个worker和1个vector来执行。
  - 内核区的指令不再被支持。
  - 递减的循环现在可以并行化。
  - 只嵌套了1个循环的内核区现在可以并行化。
  - 嵌套循环中只有最外层循环可以并行化。
  - 包含同级循环(sibling loops)的嵌套循环不支持并行化。
- 代表性地, 使用OpenACC 并行构建呈现了更好的表现,相比初识支持的OpenACC内核构建。
- device_type条款不被支持。bind和nohost条款不被支持。host_data指令在Fortran中不被支持。
- 嵌套并行化(相比于CUDA动态并行化)不被支持。
- 使用OpenACC构建内部多线程的目录(比如OpenMP或pthread programming)不被支持。
- 如果一个acc_on_device函数的调用有编译时构建，函数调用只对C和C++进行编译时常量评估但不包括Fortran。

C Family C语言家族

- OpenMP specification Version 4.5现在支持C和C++编译器。
- C和C++编译器现在支持计数器的属性。例如,现在可以支持将计数器用做声明:

```
enum {
  newval,
  oldval __attribute__ ((deprecated ("too old")))
```

};

- C和C++编译器源的位置现在可以作为范围被追踪，而不再仅仅是指针，使得识别子表达式。

例如: test.cc: 在函数'int test(int, int, foo, int, int)'中:

- test.cc:5:16: error: no match for 'operator*' (operand types are 'int' and 'foo')

```
return p + q * r * s + t;
           ~~^~~
```

- 另外, 现在开始支持精确诊断字符串中的位置: format-strings.c:3:14: warning: field width specifier '*' expects a matching 'int' argument [-Wformat=]

```
printf("%*d");
 ^
```

- 诊断器现在包含"修改提示(fix-it hints)",其显示在相关源代码的下方。例如: this is currently the only example in the tree; various others are pending fixits.c: In function 'bad_deref':
  fixits.c:11:13: error: 'ptr' is a pointer; did you mean to use '->'?

```
 return ptr.x;
        ^
```

- C和C++编译器现在提供误拼字段名的建议:
  spellcheck-fields.cc:52:13: error: 'struct s' has no member named 'colour'; did you mean 'color'?

```
return ptr->colour;
         ^~~~~~
```

- C和C++ 编译器有了新的命令行选项:
  - -Wshift-negative-value警告一个负值的左移。
  - -Wshift-overflow警告一个左移溢出，这个警告默认有效。
    -Wshift-overflow=2同样警告一个左移的1进入有符号位。

- -Wtautological-compare警告自比较总是true或者false。这个警告通过-Wall生效。
- -Wnull-dereference warns警告编译器发现的由于未关联指针产生的导致错误的或者未定义的路径。该功能只有-fdelete-null-pointer-checks激活才被激活，其被大多数的目标优化支持。警告的精确性取决于所使用的优化选项。
- -Wduplicated-cond warns警告if-else-if链中的重复条件。
- -Wmisleading-indentation warns警告对读代码的人产生误解的代码块结构。例如， CVE-2014-1266: **sslKeyExchange.c: In function 'SSLVerifySignedServerKeyExchange':**
- **sslKeyExchange.c:629:3: warning:** this **'if'** clause does not guard... **[-Wmisleading-indentation]**

```
    if ((err = SSLHashSHA1.update(&hashCtx, &signedParams)) != 0)
    ^~
```

- **sslKeyExchange.c:631:5: note:** ...this statement, but the latter is misleadingly indented as if it is guarded by the **'if'**

```
        goto fail;
        ^~~~
```

- 该警告可通过-Wall激活。

.....

New Targets and Target Specific Improvements新的架构以及架构的优化

AArch64

- 一系列特定于AArch64的操作被添加。最重要的就是如下的这些：
- 新的命令行指令-march=native,-mcpu=native,-mtune=native现在可在AArch64 GNU/Linux原生系统上使用。这些指令的细化使得GCC可以自动侦测主机CPU并且选择针对系统最优的设置。
- -fpic现在当为小代码模块(-mcmodel=small)生成代码时支持。global offset table (GOT)的大小在LP64 SysV ABI系统中限制在28KiB之下,在ILP32 SysV ABI系统中限制在15KiB之下。
- AArch64端口现在支持目标属性和注释。
- 跨不同目标翻译单元的链接时优化现在被支持。

- -mtls-size=现在被支持。其可以用来细化TLS偏移的位的大小,使得GCC可以生成更好的TLS指令序列。
- -fno-plt现在功能完全。
- ARMv8.1-A架构和The Large System Extensions现在被支持。它们可通过-march=armv8.1-a指令来使用。另外，+lse指令的扩展可以在类型形似的其他指令扩展中使用。The Large System提供了新的用在实现原子操作的指令集。

ARM

- ARM架构修复ARMv4t的支持被否定，将在未来GCC的版本中被移除。-mcpu和-mtune中被否定的有: arm2, arm250, arm3, arm6, arm60, arm600, arm610, arm620, arm7, arm7d, arm7di, arm70, arm700, arm700i, arm710, arm720, arm710c, arm7100, arm7500, arm7500fe, arm7m, arm7dm, arm7dmi, arm8, arm810, strongarm, strongarm110, strongarm1100, strongarm1110, fa526, fa626。arm7tdmi仍被支。-march中被否定的有:
  armv2,armv2a,armv3,armv3m,armv4。
- ARM端口现在支持目标属性和注释。
- 以下的处理器将被纳入支持(GCC identifiers in parentheses): ARM Cortex-A32 (cortex-a32), ARM Cortex-A35 (cortex-a35). The GCC identifiers can be used as arguments to the -mcpu or -mtune options, for example: -mcpu=cortex-a32 or -mtune=cortex-a35。

Heterogeneous Systems Architecture多种类系统架构

- GCC现在可以通过--enable-offload-targets=hsa为简单的OpenMP设备生成HSAIL (Heterogeneous System Architecture Intermediate Language多种类系统架构媒介语言)。一个新的libgomp插件将会使用在HSA GPU内核中以通过标准的HAS运行时实现在HAS可用GPUs中的这些构建。如果HAS编译最终显示其不能为特定的输入来输出HSAIL，其将默认发出警告。这些警告可通过-Wno-has抑制。

.....

Operating Systems操作系统

AIX

- DWARF debugging对AIX 7.1的支持已经被一个可选的debugging格式来使用。一个更新的Technology Level (TL) 和这个级别的GCC对于利用所有的DWARF debugging功能都是需要的.

Linux

- 对musl C library的支持现在被加入到AArch64, ARM, MicroBlaze, MIPS, MIPS64, PowerPC, PowerPC64, SH, i386, x32和x86_64中。当musl不是默认的libc情况下，可以通过选择使用新的命令-mmusl。如果是建立在匹配*-linux-musl*的模式的三个架构之一，GCC默认使用musl libc。

RTEMS

- RTEMS线程模式的实现方式被改变。Mutexes现在可以使用在Newlib <sys/lock.h>中定义的自约束的对象，而不是Classic API信号。线程特殊数据的键和once函数直接通过<pthread.h>定义。自约束的条件变量通过Newlib <sys/lock.h>提供。RTEMS线程模型同样支持C++11线程。
- OpenMP 现在通过Newlib <sys/lock.h>的提供来支持自约束对象，并且相比于POSIX的指令libgomp提供了一个显著提升的表现。<sys/lock.h>。

Solaris

- Solaris 12现在被全面支持。最低限度的支持已经在GCC 5.3中提供。
- Solaris 12提供了一个完全的安装文件集(crt1.o, crti.o, crtn.o), GCC相比于其自带的，更青睐这个。
- Position independent executables (PIE)现在在Solaris 12上支持。
- 构造函数的优先性现在伴随着在系统链接器在Solaris 12上支持。

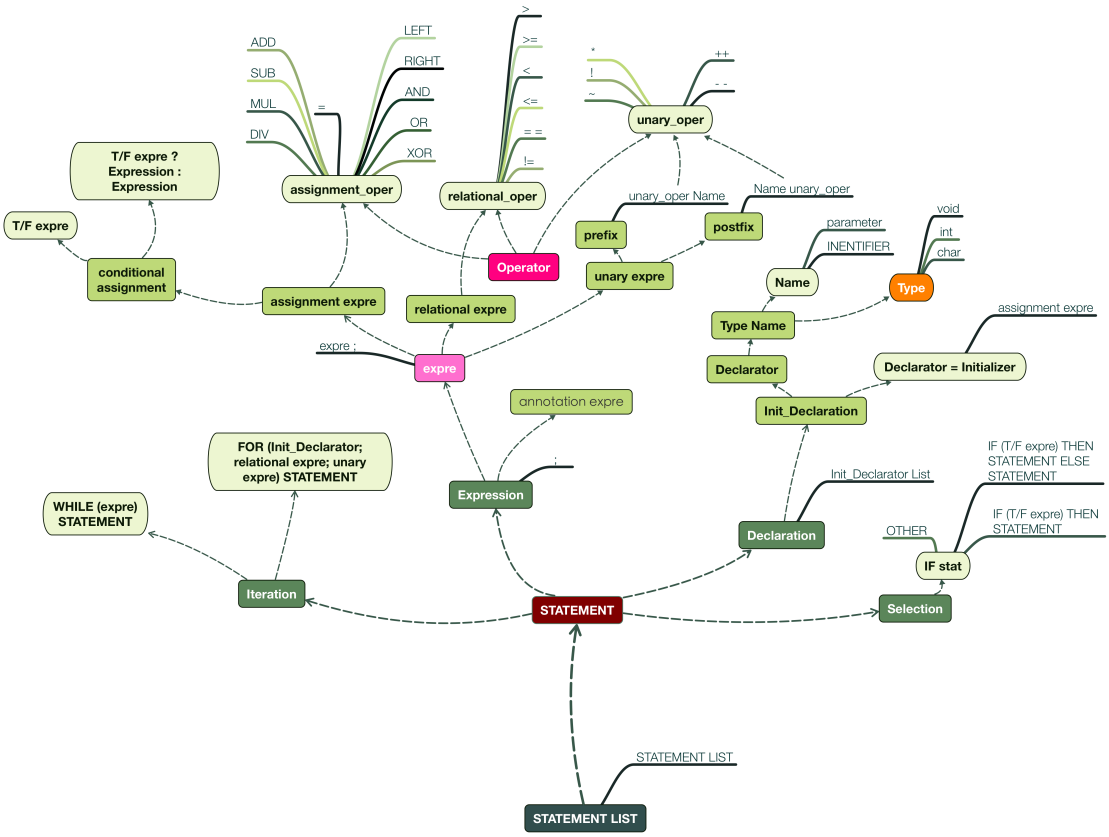
Windows

- -mstackrealign现在当使用了SSE指令时自动在32位模式中激活。

.....

五、定制自制 C 编译器支持语言子集

我通过一张思维导图的形式更清晰地表达了所支持子集的上下文无关文法关系：



图：自绘【上下文无关文法树】

Type →

void

| char

| int

Name→

IDENTIFIER

|parameter

Declarator→

Type Name

Declaration→

Init_Declaration

initializer →

assignment expre

STATEMENT LIST→

STATEMENT

| STATEMENT LIST

STATEMENT →

Iteration

| Expression

| Declaration

| Selection

Expression →

expre ;

| annotation expre ;

| ;

IF stat →

IF (T/F expre) THEN STATEMENT ELSE STATEMENT

| IF (T/F expre) THEN STATEMENT

Iteration →

WHILE (T/F expre) STATEMENT

|FOR (Init_Declarator; relational expre; unary expre) STATEMENT

expre →

assignment expre

| relational expre

| unary expre

| expre

assignment expre →

conditional assignment

| Name assignment oper Name

conditional assignment →

T/F expre

| T/F expre ? Expression : Expression

Operator →

assignment oper

| relational oper

| unary oper

unary expre

prefix

| postfix

prefix →

unary_oper Name

postfix →

Name unary_oper

unary_oper →

*

| +

| -

| ~

| !

assignment oper →

=

| MUL

| DIV

| ADD

| SUB

| LEFT

| RIGHT

| AND

| XOR

| OR

relational oper →

= =

| >

| <

| >=

| <=

| !=

六、语言子集的汇编语言描述

以下为我现阶段通过阅读课本进行的思考：

首先试着书写两种语句较为通用情况的汇编伪代码

1. IF 语句

C 描述：

IF (eax < ebx) || (eax == ecx)

THEN <logic code 1>

ELSE <logic code 2>

汇编描述：

```
if:
    cmp %eax, %ebx
    jle else
    cmpl %eax, %ecx
    jne else
then:
    <logic code 1>
    jmp end
else:
```

```
    <logic code 2>
end:
    …
```

2. FOR 语句

汇编描述：

```
for:
    <condition to evaluate for loop counter value>
    jxx forcode     ; jump to the code of the condition is true
    jmp end         ; jmp to the end if the condition is false
forcode:
    <for loop code to execute>
    <increment for loop counter>
    jmp for         ; go back to the start of the FOR statement
end:
    …
```

编译器在分析阶段把一个源程序划分成各个组成部分，并生成源程序的内部表示（中间代码）。然后，编译器在合成阶段将这个中间代码翻译成目标程序。

构造语法制导翻译器要从原语言的文法开始。在描述一个翻译器时，在程序构造中附加属性非常有用。词法分析器从输入中逐个读入字符，应输出一个此法单元的流。翻译器使用符号表来存放保留字和已经遇到的标识符。语法分析指出如何从一个文法的开始符号推导出一个给定的终结符号串。使用预测分析方法（自顶向下）可以高效建立语法分析器。语法分析得到中间代码。最后代码生成器以编译器前端生成的中间表示和相关的符号表信息作为输入，输出语义等价的目标（汇编）程序。

七、写在最后

感谢李忠伟老师、王刚老师以及两位助教学长对本次实验的悉心指导和耐心解答。