编译器构造预备 实验二报告

姓名：刘嘉洋

学号：1412620

专业：软件工程

CONTENTS

自制编译器

文法表示

GCC调研报告


一、GCC调研

GCC 6 Release Series
Changes, New Features, and Fixes

This page is a brief summary of some of the huge number of improvements in GCC 6. For more information, see the Porting to GCC 6 page and the full GCC documentation.

Caveats

- The default mode for C++ is now `-std=gnu++14` instead of `-std=gnu++98`.
- Support for a number of older systems and recently unmaintained or untested target ports of GCC has been declared obsolete in GCC 6. Unless there is activity to revive them, the next release of GCC will have their sources permanently **removed**.

  The following ports for individual systems on particular architectures have been obsoleted:

  - SH5 / SH64 (sh64-*-*) as announced here.
- The AVR port requires binutils version 2.26.1 or later for the fix for PR71151 to work.

General Optimizer Improvements

- UndefinedBehaviorSanitizer gained a new sanitization option, `-fsanitize=bounds-strict`, which enables strict checking of array bounds. In particular, it enables `-fsanitize=bounds` as well as instrumentation of flexible array member-like arrays.
- Type-based alias analysis now disambiguates accesses to different pointers. This improves precision of the alias oracle by about 20-30% on higher-level C++ programs. Programs doing invalid type punning of pointer types may now need `-fno-strict-aliasing` to work correctly.
- Alias analysis now correctly supports `weakref` and `alias` attributes. This makes it possible to access both a variable and its alias in one translation unit which is common with link-time optimization.
- Value range propagation now assumes that the `this` pointer of C++ member functions is non-null. This eliminates common null pointer checks but also breaks some non-conforming code-bases (such as Qt-5, Chromium, KDevelop). As a temporary work-around `-fno-delete-null-pointer-checks` can be used. Wrong code can be identified by using `-fsanitize=undefined`.
- Link-time optimization improvements:
  - `warning` and `error` attributes are now correctly preserved by declaration linking and thus `-D_FORTIFY_SOURCE=2` is now supported with `-flto`.
  - Type merging was fixed to handle C and Fortran interoperability rules as defined by the Fortran 2008 language standard.

    As an exception, `CHARACTER(KIND=C_CHAR)` is not inter-operable with `char` in all cases because it is an array while `char` is scalar. `INTEGER(KIND=C_SIGNED_CHAR)` should be used instead. In general, this inter-operability cannot be implemented, for example, on targets where function passing conventions of arrays differs from scalars.
  - More type information is now preserved at link time reducing the loss of accuracy of the type based alias analysis compared to builds without link-time optimization.
  - Invalid type punning on global variables and declarations is now reported with `-Wodr-type-mismatch`.
  - The size of LTO object files was reduced by about 11% (measured by compiling Firefox 46.0).
  - Link-time parallelization (enabled using `-flto=n`) was significantly improved by decreasing the size of streamed data when partitioning programs. The size of streamed IL while compiling Firefox 46.0 was reduced by 66%.
  - The linker plugin was extended to pass information about type of binary produced to GCC back end (that can be also manually controlled by `-flinker-output`). This makes it possible to properly configure the code generator and support incremental linking. Incremental linking of LTO objects by `gcc -r` is now supported on plugin-enabled setups.


新的架构以及
架构优化
常用优化器的改进
C语言家族
新的语言以及
语言优化

- 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。
- **程间优化**(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。

- **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CHARACTER(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的后台(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。有两种方式执行增量链接:
  - 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。
  - 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。

相比GCC 5, GCC 6发布系列亮点包括:

- 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。

相比GCC5,GCC6“不再支持”：

- 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。

- 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，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，而不再仅仅是点位，使得识别子表达式。例如：

```
test.cc: In function 'int test(int, int, foo, int, int)':
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:

```
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。

- *-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;
        ^~~~
```

AArch64

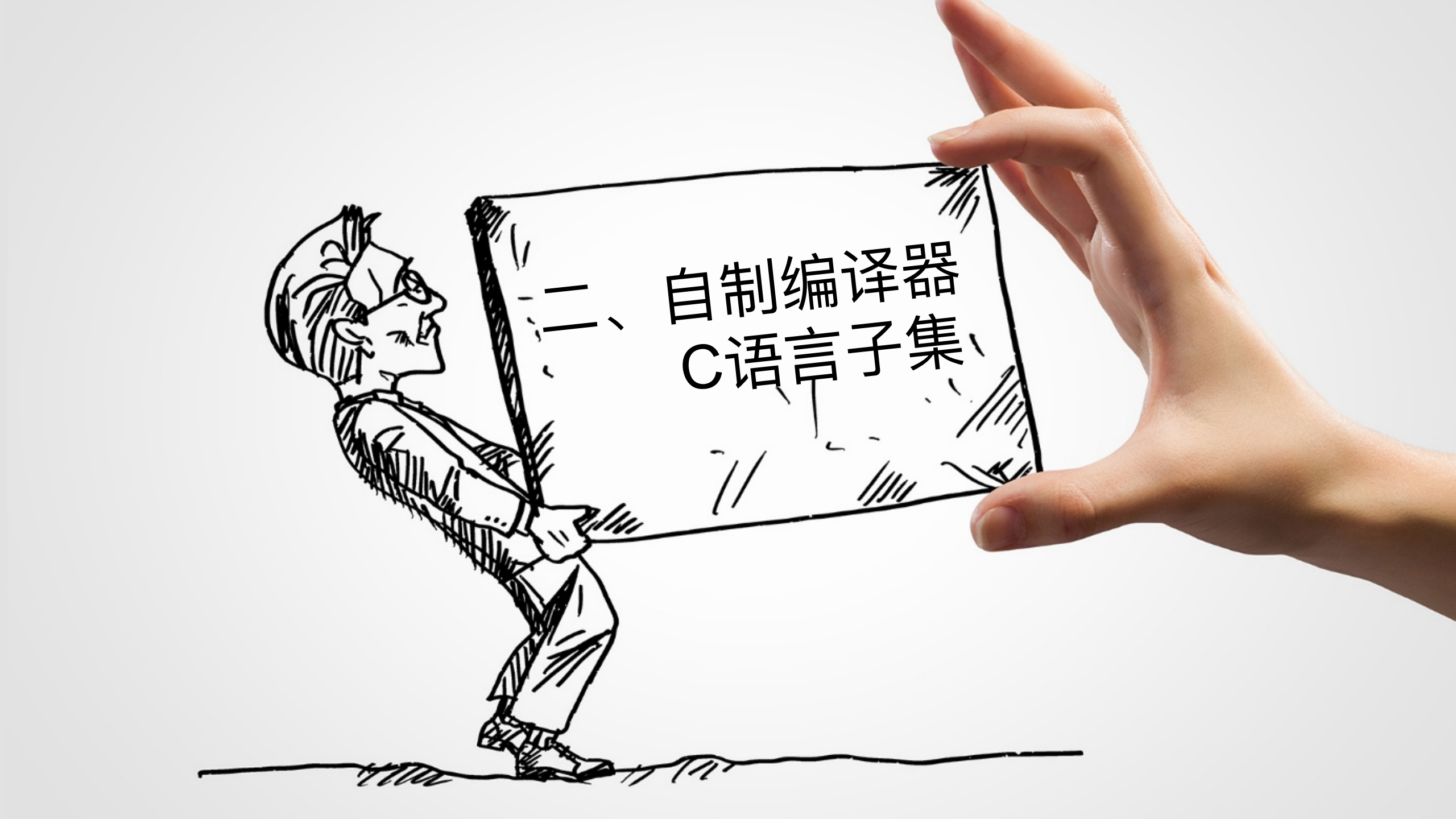
- 一系列特定于AArch64的操作被添加。最重要的就是如下的这些：
- 新的命令行指令*-march=native*,*-mcpu=native*,*-mtune=native*现在可在AArch64 GNU/Linux原生系统上使用。这些指令的细化使得GCC可以自动侦测主机CPU并且选择针对系统最优的设置。
- -fpic现在当为小代码模块(*-mcmodel=small*)生成代码时支持。global offset table (GOT)的大小在LP64 SysV ABI系统中限制在28KiB之下,在ILP32 SysV ABI系统中限制在15KiB之下。
- AArch64端口现在支持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

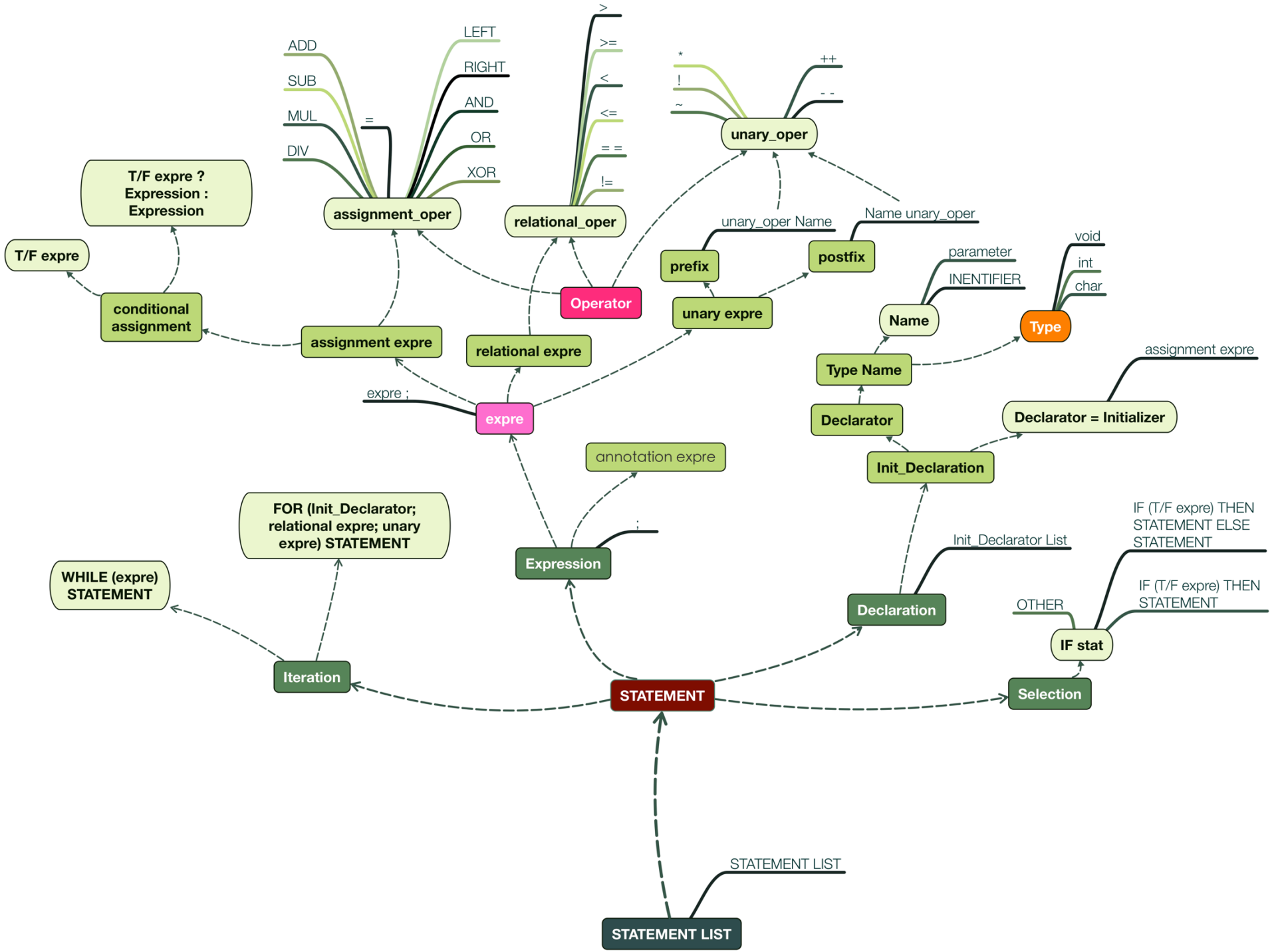
- ARM架构修复ARMv4t的支持被否定，将在未来GCC的版本中被移除。*-mcpu*和*-mtune*中被否定的有: arm2, arm250, arm3, arm6, arm60, arm600, arm610, arm620, arm7, arm7d, arm7di, arm70, arm700, arm700i, arm710, arm720, arm710c, arm7100, arm7500, arm7500fe, arm7m, arm7dm, arm7dmi, arm8, arm810, strongarm, strongarm110, strongarm1100, strongarm1110, 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。


二、自制编译器
C语言子集

上下文无关文法树


ADD
SUB
MUL
DIV
=
LEFT
RIGHT
AND
OR
XOR
assignment_oper
>
>=
<
<=
= =
!=
relational_oper
*
!
~
++
- -
unary_oper
T/F expre ?
Expression :
Expression
T/F expre
conditional
assignment
assignment expre
Operator
relational expre
unary_oper Name
Name unary_oper
prefix
postfix
unary expre
parameter
INENTIFIER
Name
void
int
char
Type
Type Name
Declarator
Declarator = Initializer
assignment expre
Init_Declaration
expre ;
expre
annotation expre
;
Expression
FOR (Init_Declarator;
relational expre; unary
expre) STATEMENT
WHILE (expre)
STATEMENT
Init_Declarator List
Declaration
IF (T/F expre) THEN
STATEMENT ELSE
STATEMENT
IF (T/F expre) THEN
STATEMENT
OTHER
IF stat
Iteration
STATEMENT
Selection
STATEMENT LIST
STATEMENT LIST


感谢聆听！
姓名：刘嘉洋
学号：1412620
专业：软件工程