

GNU toolchain

Как скомпилировать вашу программу

Семён Зернов

➤ Системы компиляции

- GNU toolchain
- Компилятор
- Ассемблер
- Линкер

Осаждённый дурдом

Языки программирования



...

Целевые архитектуры

x86



arm

MIPS

...

Задачка по комбинаторике

M – языков программирования

N – целевых архитектур

Как скомпилировать исходные коды
в ассемблеры целевых архитектур?

file.cpp

file.c

file.f
(Fortran)

file.rs
(Rust)

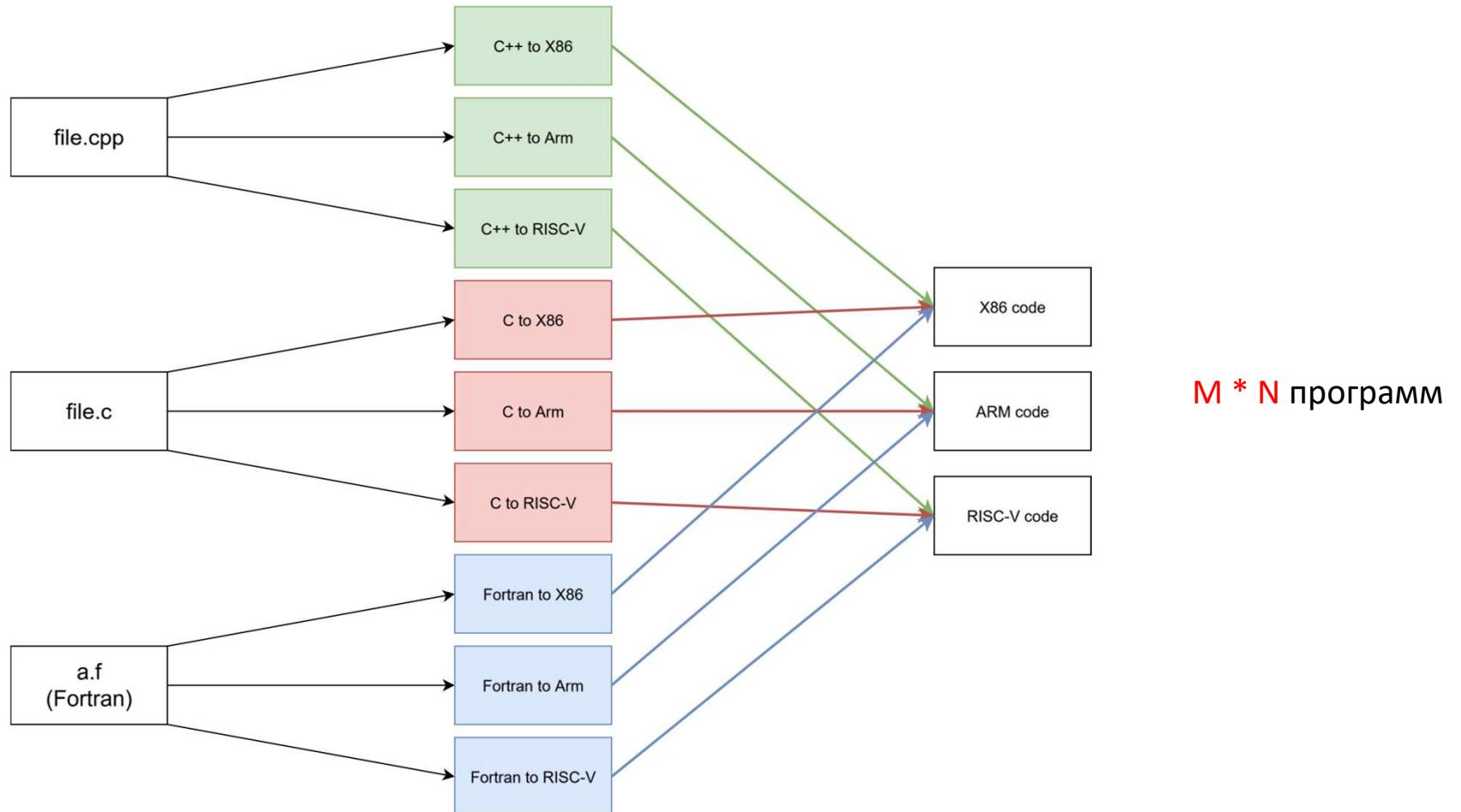
file.go
(Golang)

X86 code

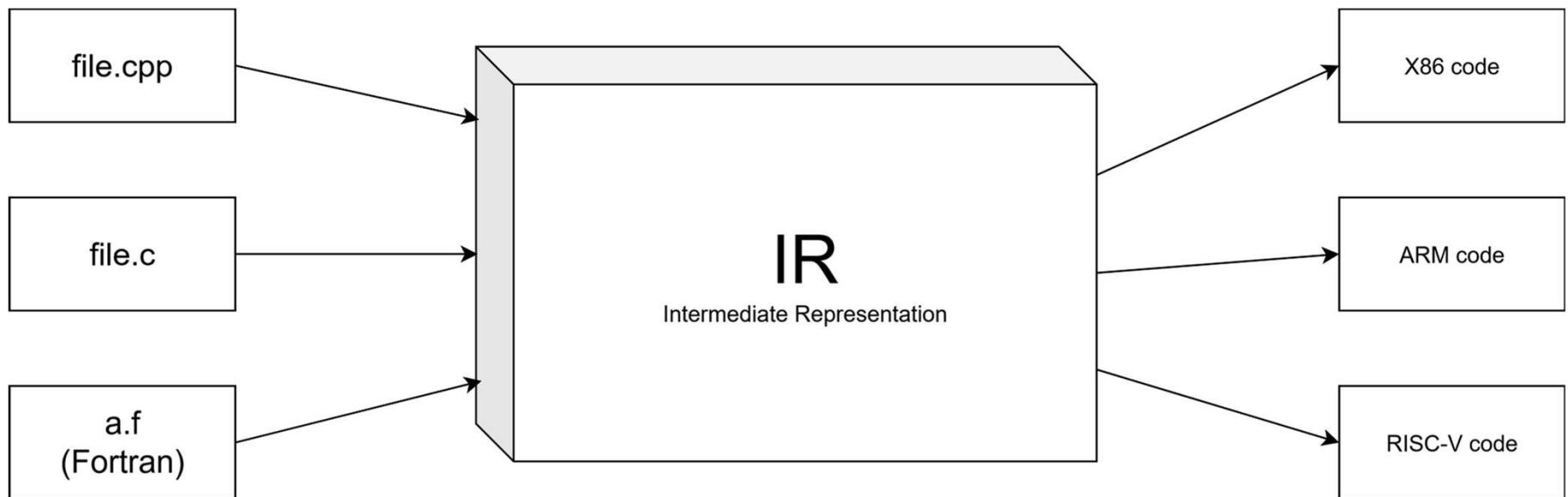
ARM code

RISC-V code

ПМ-ПУШНИКИ КОД ПИСАЛИ

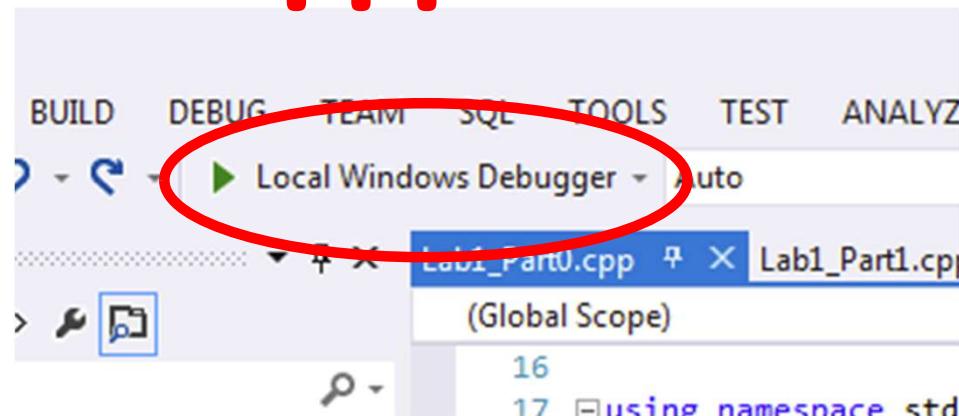


Механики делают М + Н

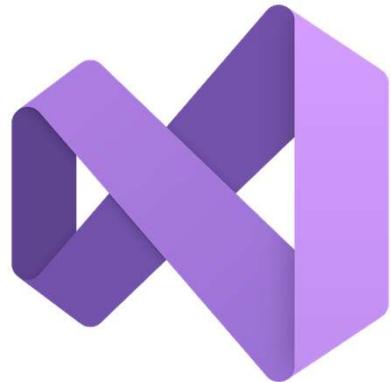


Вся наша лекция об одной кнопке

???



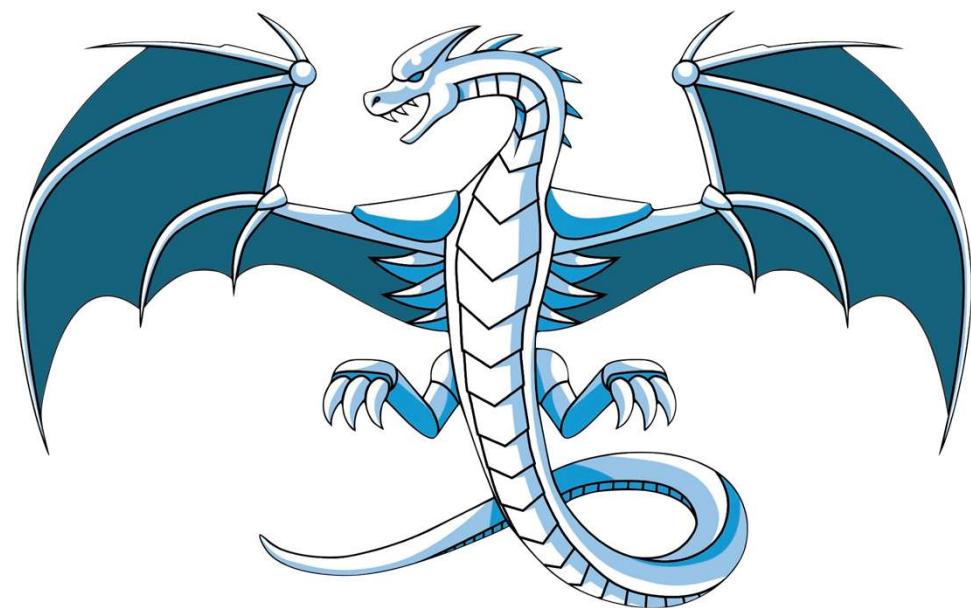
Системы компиляции



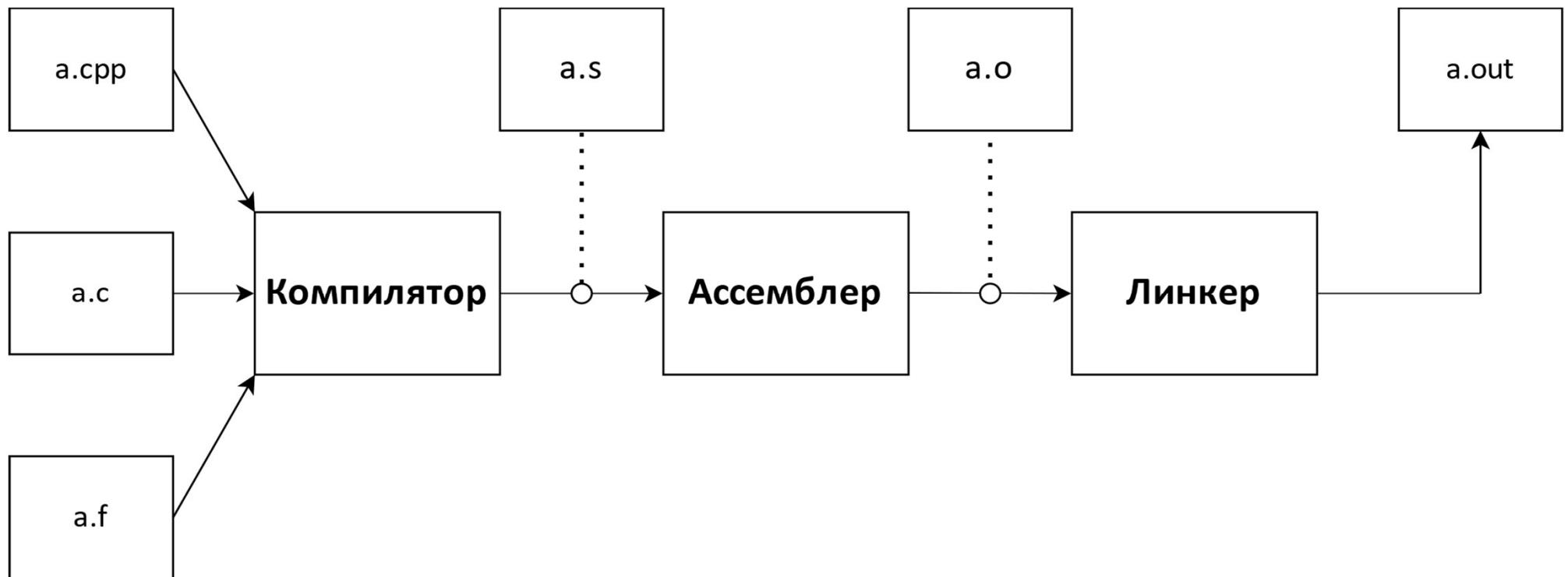
MSVC



LLVM



Системы компиляции



- Системы компиляции

- GNU toolchain

- Компилятор

- Ассемблер

- Линкер

GNU toolchain

- Авторы: Ричард Мэттью Столлман (rms) и сообщество
- ~ 15 миллионов LOC



What's bad about: [Airbnb](#) | [Amazon](#) | [Amtrak](#) | [Ancestry](#) | [Apple](#) | [Cloudflare](#) | [Discord](#) | [Ebooks](#) | [Eventbrite](#) | [Evernote](#) | [Facebook](#) | [Frito-Lay](#) | [Frontier](#) | [Google](#) | [Gofundme](#) | [Grubhub](#) | [Intel](#) | [LinkedIn](#) | [Lyft](#) | [Meetup](#) | [Microsoft](#) | [Netflix](#) | [Patreon](#) | [Pay Toilets](#) | [Skype](#) | [Slack](#) | [Spotify](#) | [Tesla](#) | [Ticketmaster](#) | [Twitter](#) | [Uber](#) | [Wendy's](#) | [WhatsApp](#) | [Zoom](#)

[RSS site feed](#) for the most recent political notes and new material.

This is the personal web site of Richard Stallman.

The views expressed here are my personal views, not those of the [Free Software Foundation](#) or the [GNU Project](#).

For the sake of separation, this site has always been hosted elsewhere and managed separately.

If you want to send me GPG-encrypted mail, do not trust key servers! Some of them have phony keys under my name and email address, made by someone else as a trick. See [gpg.html](#) for my real key.

Don't watch TV coverage of Covid-19!

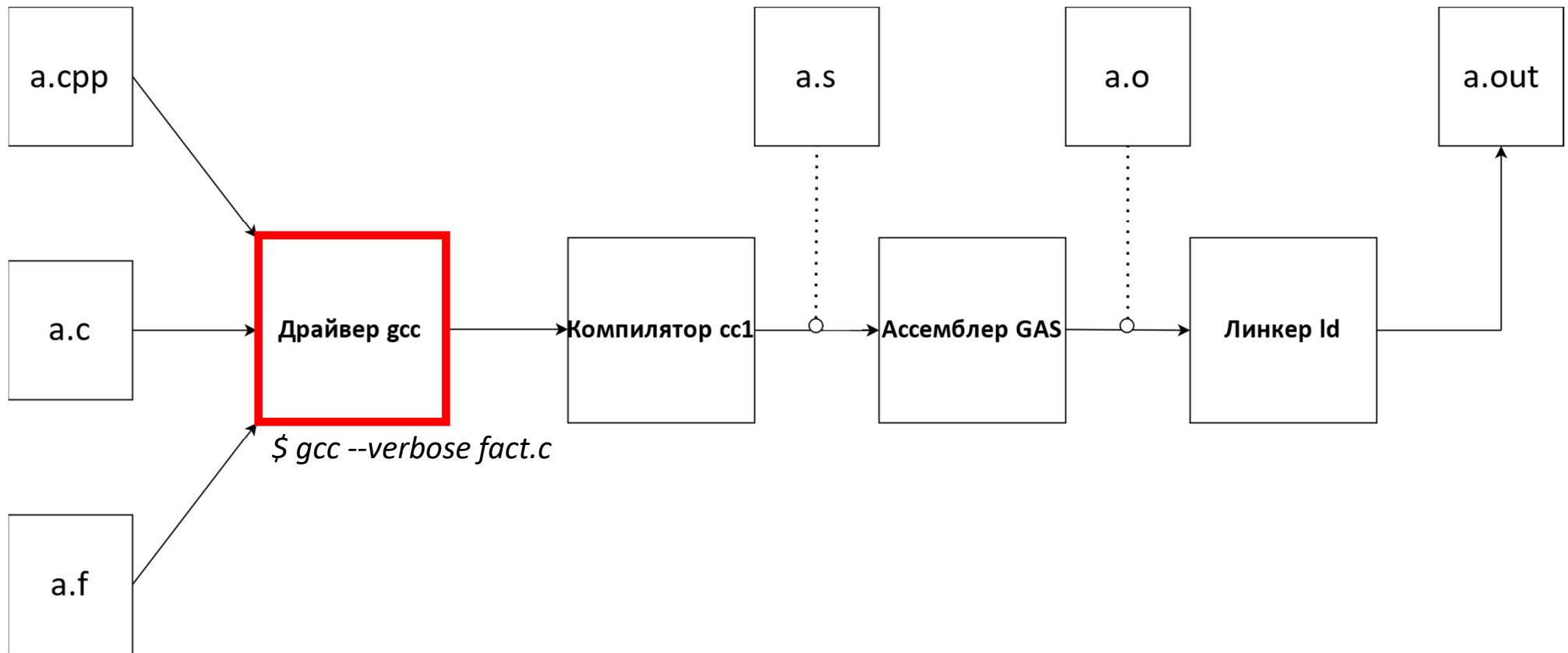
GNU toolchain

- gcc — компилятор (программа-драйвер, на самом деле компилятор называется cc1)
- make — система сборки
- GAS — ассемблер
- ld — линкер
- gdb — отладчик
- perf — анализатор производительности
- objdump — парсер объектных файлов
- ...



[GNU Binutils](#)

GNU toolchain



- Системы компиляции

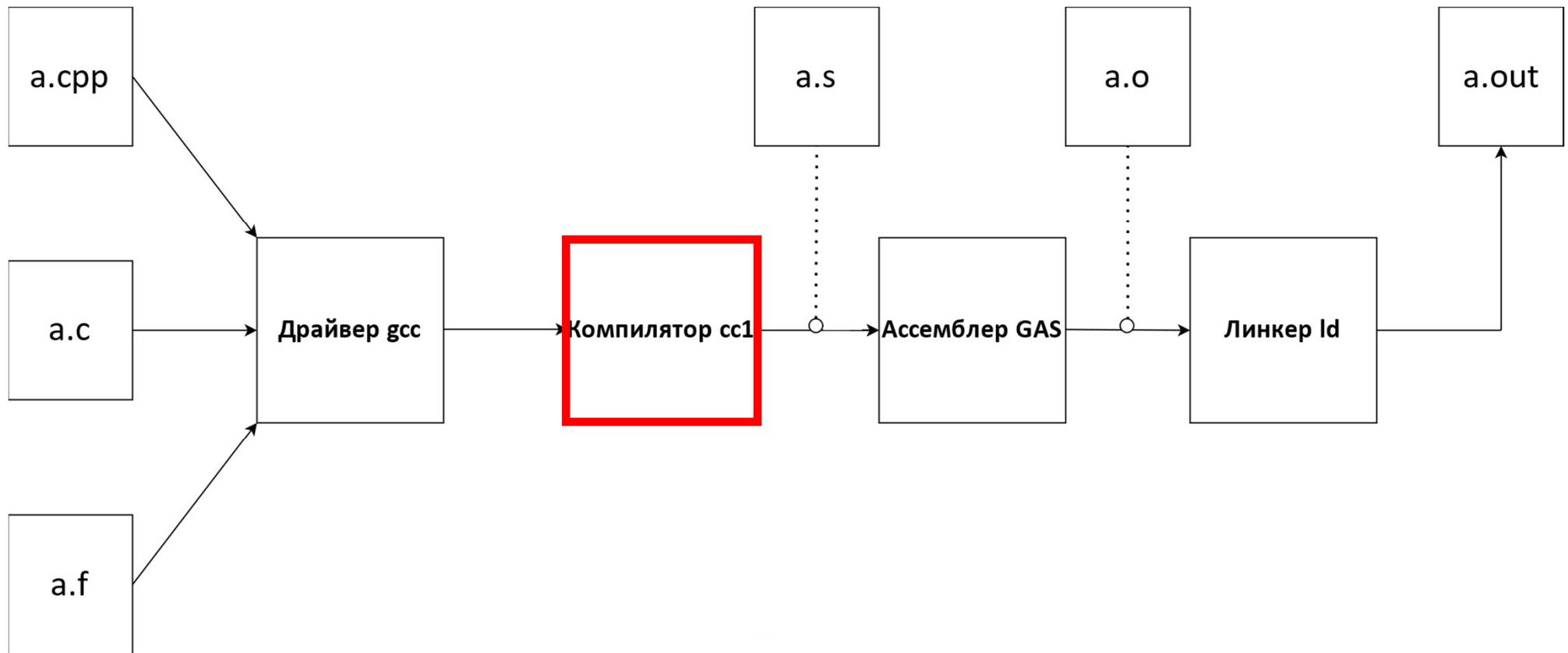
- GNU toolchain

- Компилятор

- Ассемблер

- Линкер

GNU toolchain



Примитивнейшая схема компилятора

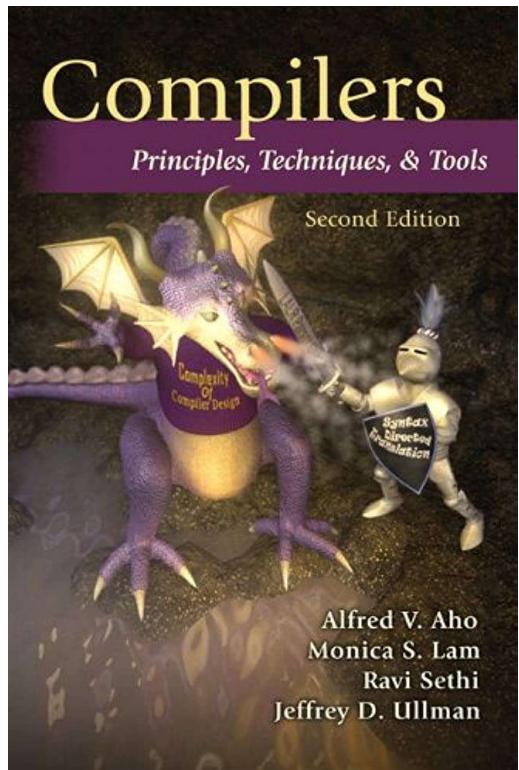


Frontend

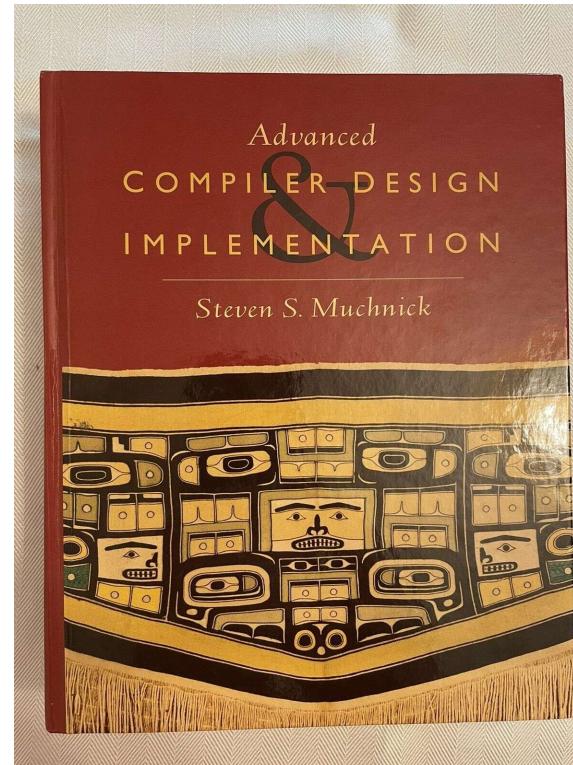
MiddleEnd/Backend

Компилятор

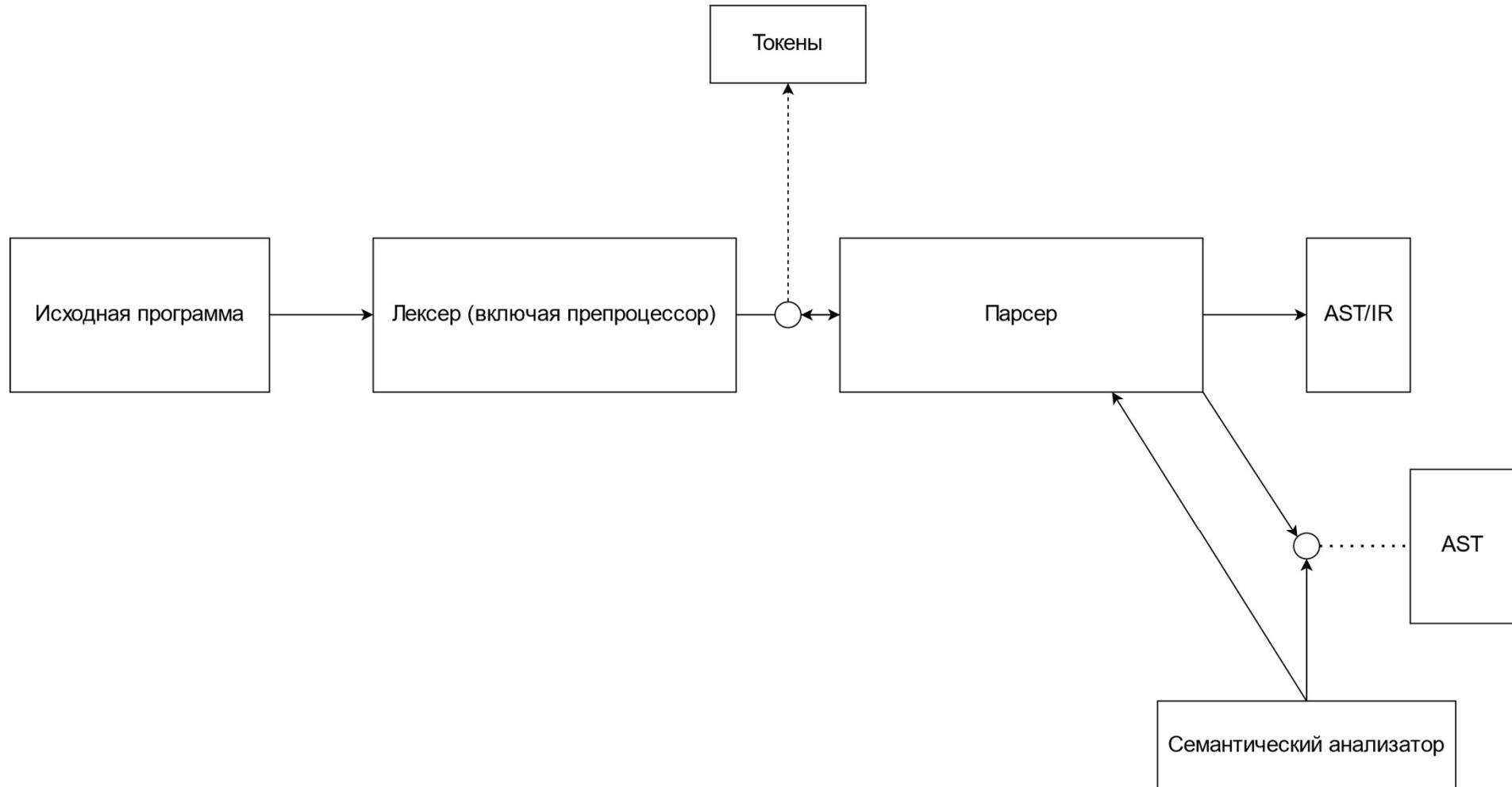
Frontend



MiddleEnd/Backend



Frontend



Лексер

Преобразует программу в последовательность лексем (tokens). Производит лексический анализ — проверка свойств регулярных языков с помощью ДКА.

Типичная ошибка: \$ в имени переменной.

- Генерация preprocessing tokens
- Дальше включается препроцессор (подстановка include-ов, раскрытие макросов)
- Повторяем, пока не останется нераскрытых макросов

Результат работы лексера: чистые лексемы (или ошибка, при лексических ошибках)

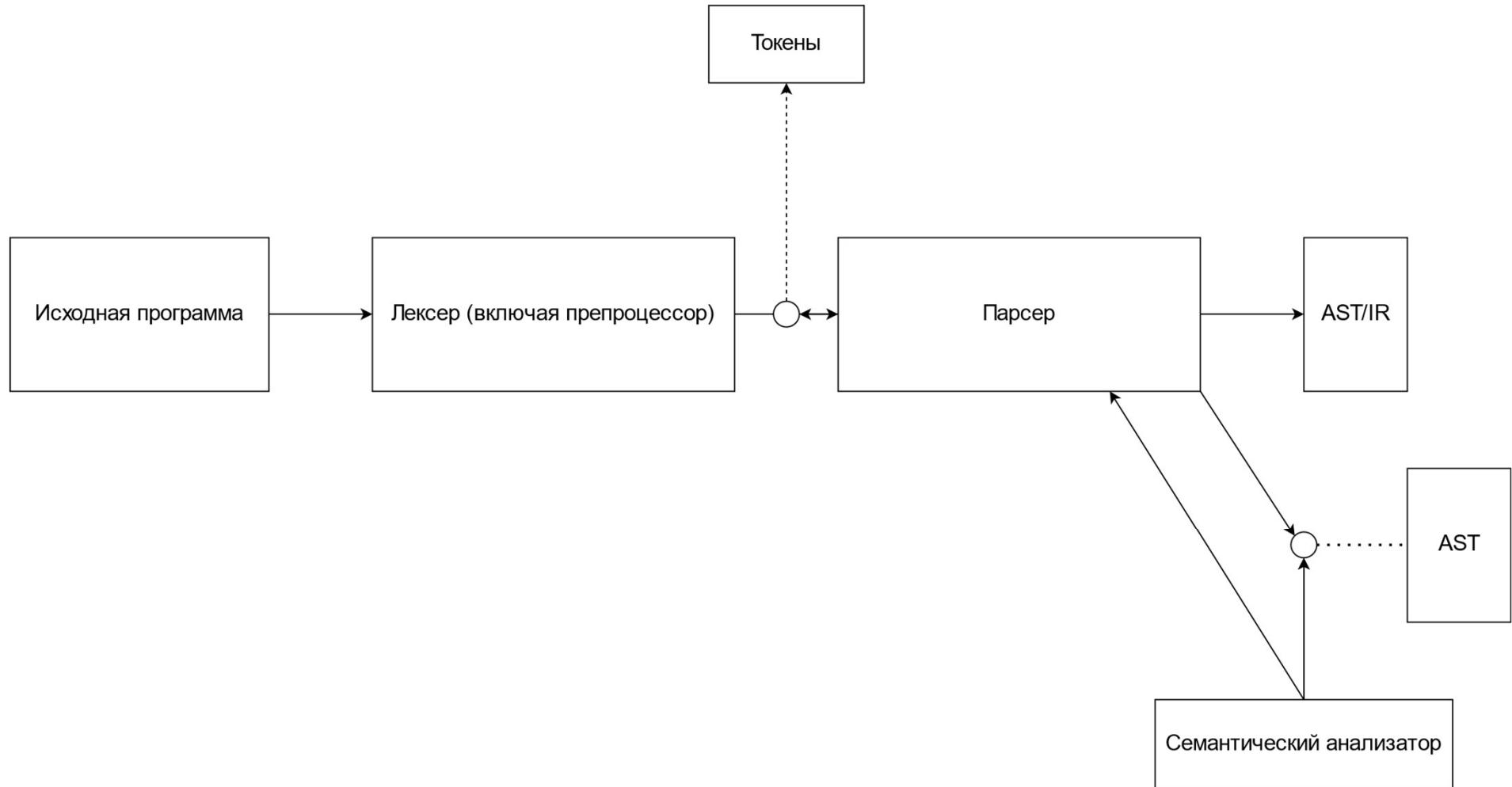
```
int main (
//hello
) {
return 0
; }
```

Лексер

See: [C++ Translation Phases](#)



Frontend



Синтаксический и семантический анализ

Парсер производит синтаксический анализ (проверка свойств контекстно-свободных грамматик с помощью рекурсивного спуска)

Модуль семантического анализа: проверка свойств контекстно-зависимых грамматик, самая сложная часть

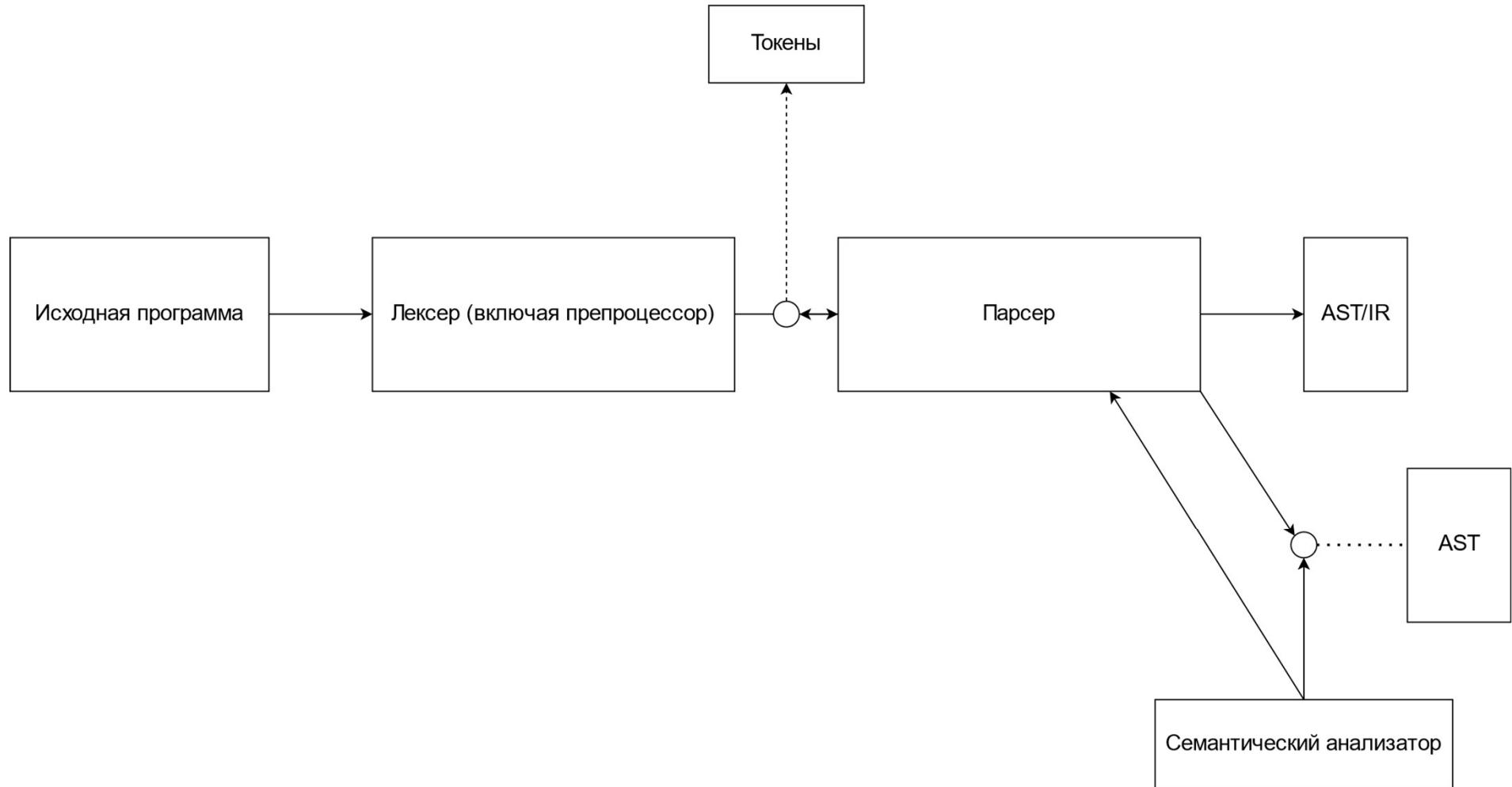
Синтаксическая ошибка:
((())() - незакрытая скобка

Семантическая ошибка:
`int foo(int a, int b);
foo(5);`
несовпадение аргументов функции при вызове

Во время работы парсер и семантический анализатор вызывают друг друга



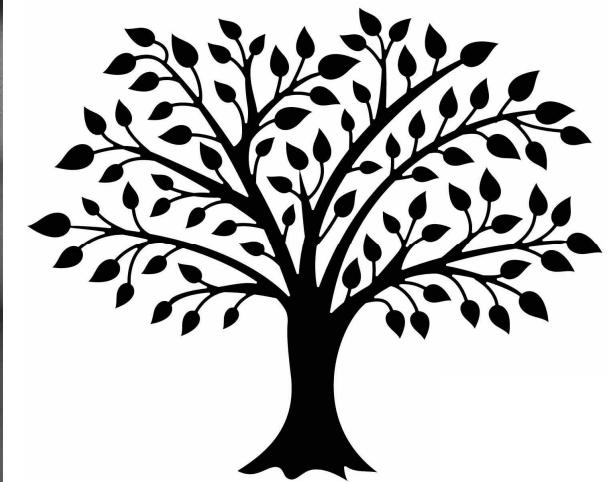
Frontend



AST



\$ clang -Xclang -ast-dump
fact.c



IR

В GCC существуют следующие IR:

- GENERIC
- gimple
- gimple SSA
- RTL (virtual registers)
- RTL (physical registers) – после regalloc



SSA — static
single assignment

Оптимизации

```
$ gcc -O2 -fdump-rtl-all -fdump-tree-all fact.c -o fact
```

Современные компиляторы проводят над кодом множество оптимизаций. Основные группы:

- Machine independent, например: inline, recursion elimination
- Machine dependend, например: scheduler



[Optimizations list](#)

Работа окончена... или нет?

```
$ gcc -O2 -S fact.c -o fact.s
```

- Результат работы компилятора - ассемблер конкретной архитектуры (x86, ARM, RISC-V)
- Можно ли его уже запустить? Конечно, нет.

Передаём работу ассемблеру



- Системы компиляции

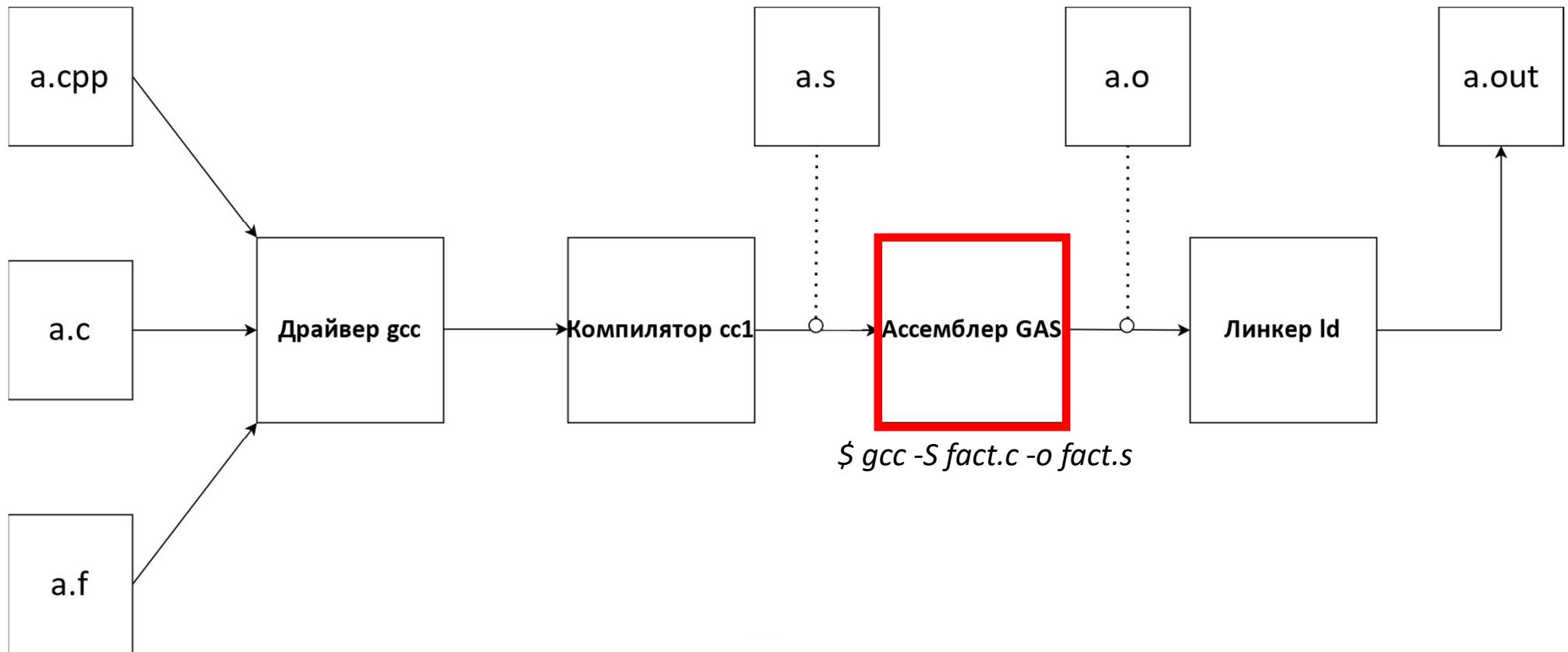
- GNU toolchain

- Компилятор

- Ассемблер

- Линкер

GNU toolchain



Ассемблер

Это слово многозначно:

- Язык программирования
- Программа, производящая *ассемблирование*



... И ЧТО В ИТОГЕ?

- Результат работы ассемблера — объектный файл (`fact.o`)
- Мы собрали в объектный файл одну единицу трансляции (грубо говоря, один файл `fact.c`)
- Реальные проекты состоят из сотен Translation Unit, и всех их надо собрать в один бинарник



- Системы компиляции

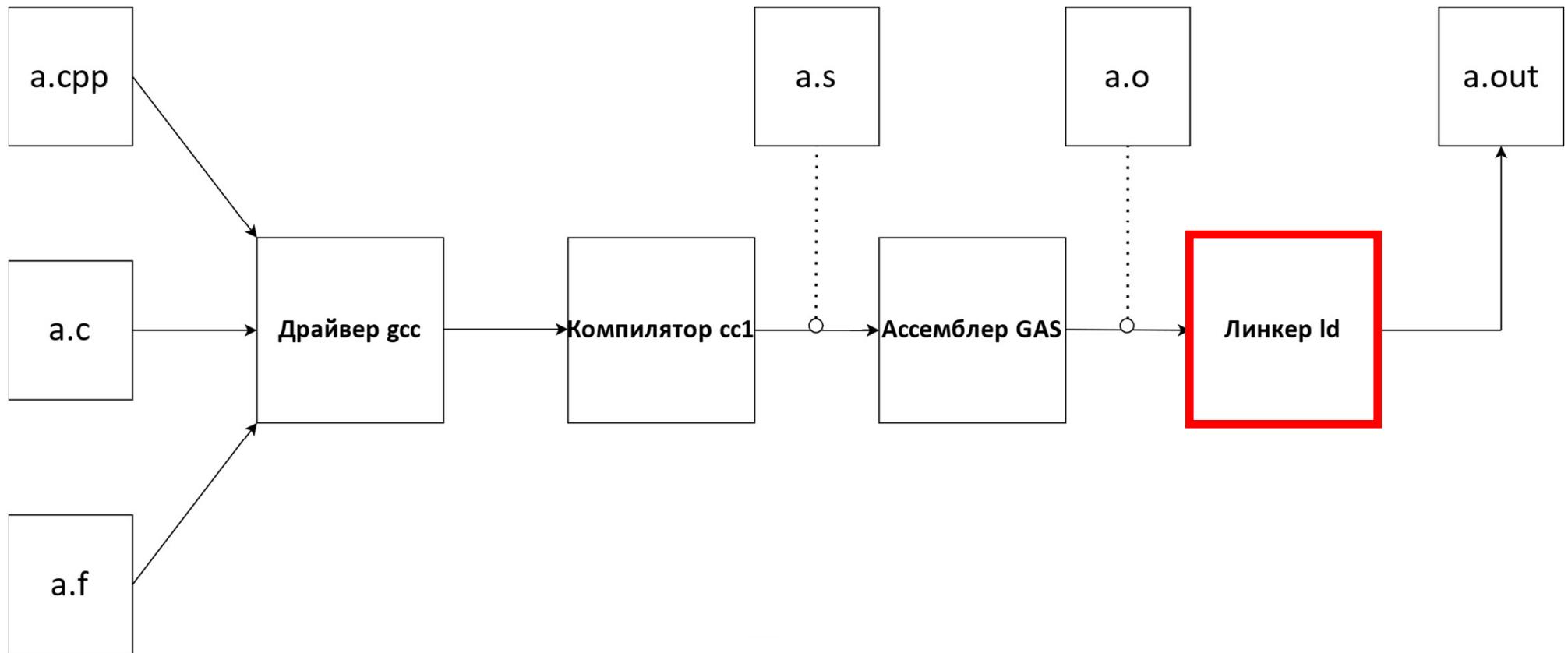
- GNU toolchain

- Компилятор

- Ассемблер

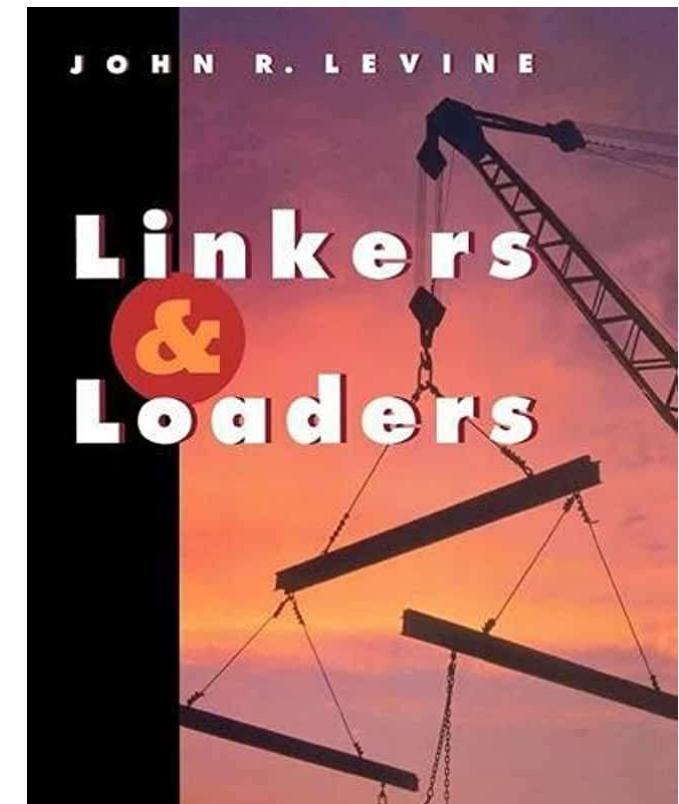
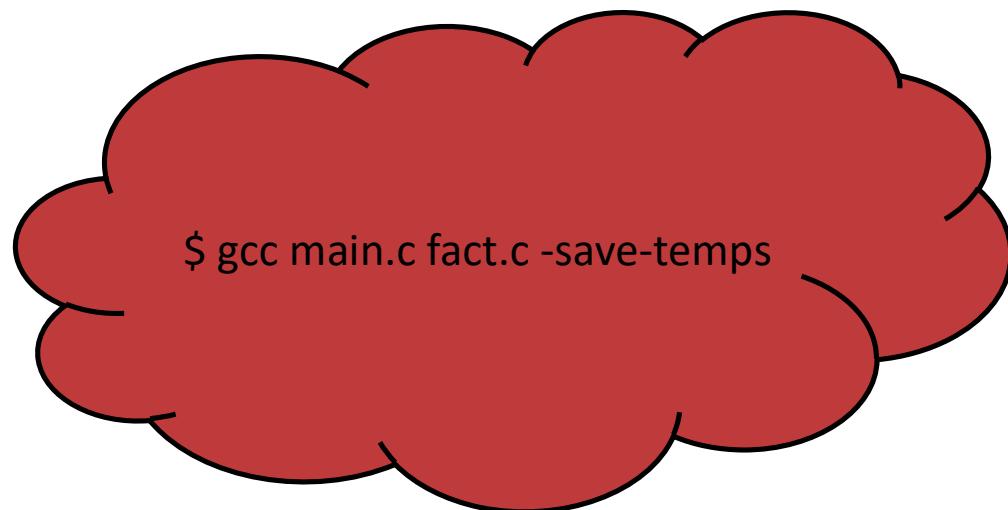
- Линкер

GNU toolchain



Линкер

Слинкуем простую программу



Список литературы

- [GCC manual](#)
- Константин Владимиров, [Toolchain & pony, 2020](#)
- [RMS homepage](#)
- [Aho, Lam, Sethi, Ullman], Compilers: Principles, Techniques, and Tools, 2006
- Steven Muchnik, Advanced compiler design and implementation, 1997
- Jonh Levine, Linkers and Loaders, 1999

Избранным

Sanitizers



[Sanitizers](#)