LLVM Hands On

Internal of LLVM and what we can get from

Henri-Pierre Charles

Université de Versailles Saint-Quentin en Yvelines

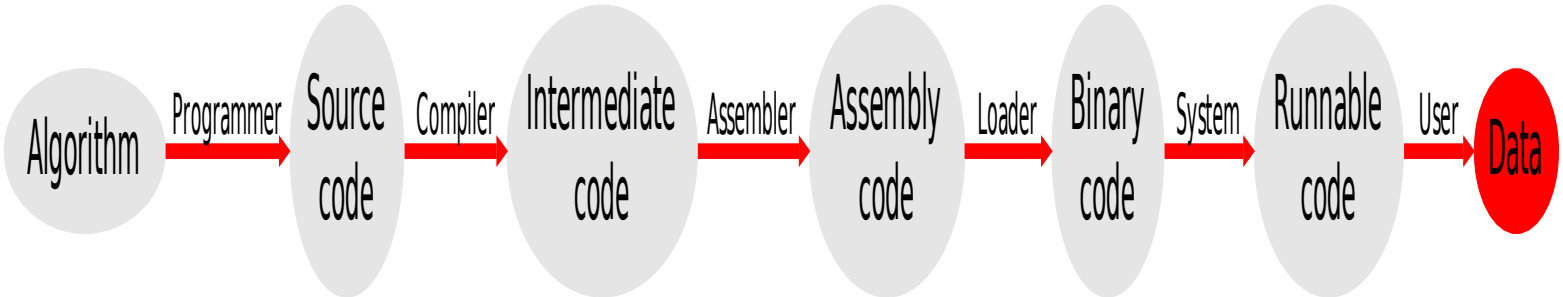
23 novembre 2009

Présentation

- Université de Versailles
- Laboratoire PRiSM
- Optimisation de code
- Partenariats académiques industriels : CEA, Intel, Bull

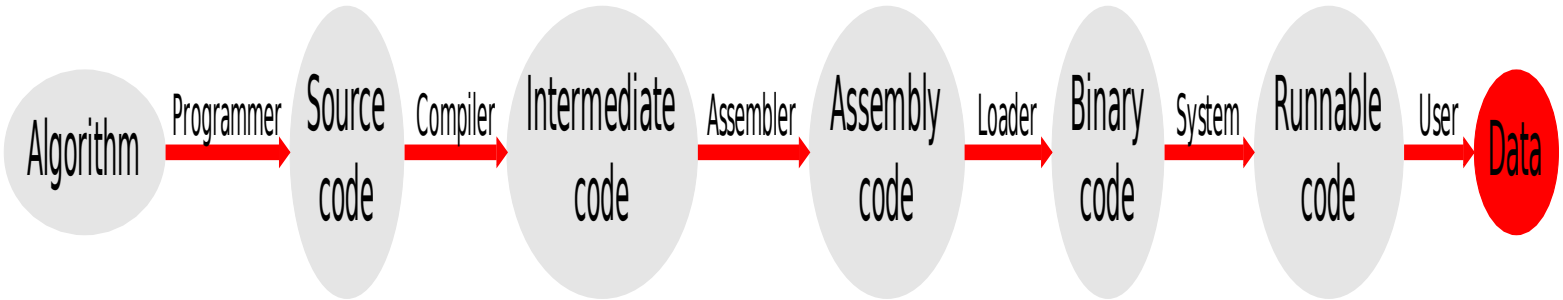
Le defaut de performance est devenu un bug

What's a compiler ?



- Historical point of view :
  - Translate a high level language to the binary language
  - Correctness in mind
- Actual point of view
  - Translate a high level language to the binary language
  - Speed in mind (target code and compil time)

Chaine de compilation



Source code assumed "high level"

Exec code assumed "low level"

Optimisation assumed "data independant"

target processor assumed "known" (cell phone ?, gpu ?)

Nothing true nowadays !

What can we ask to a compiler ?

1980 produce correct code

2009

1. produce a fast code
2. automatic parallelization (multi CPU / multi core / multi thread)
3. vectorization
4. use multimedia instruction
5. instrument the code

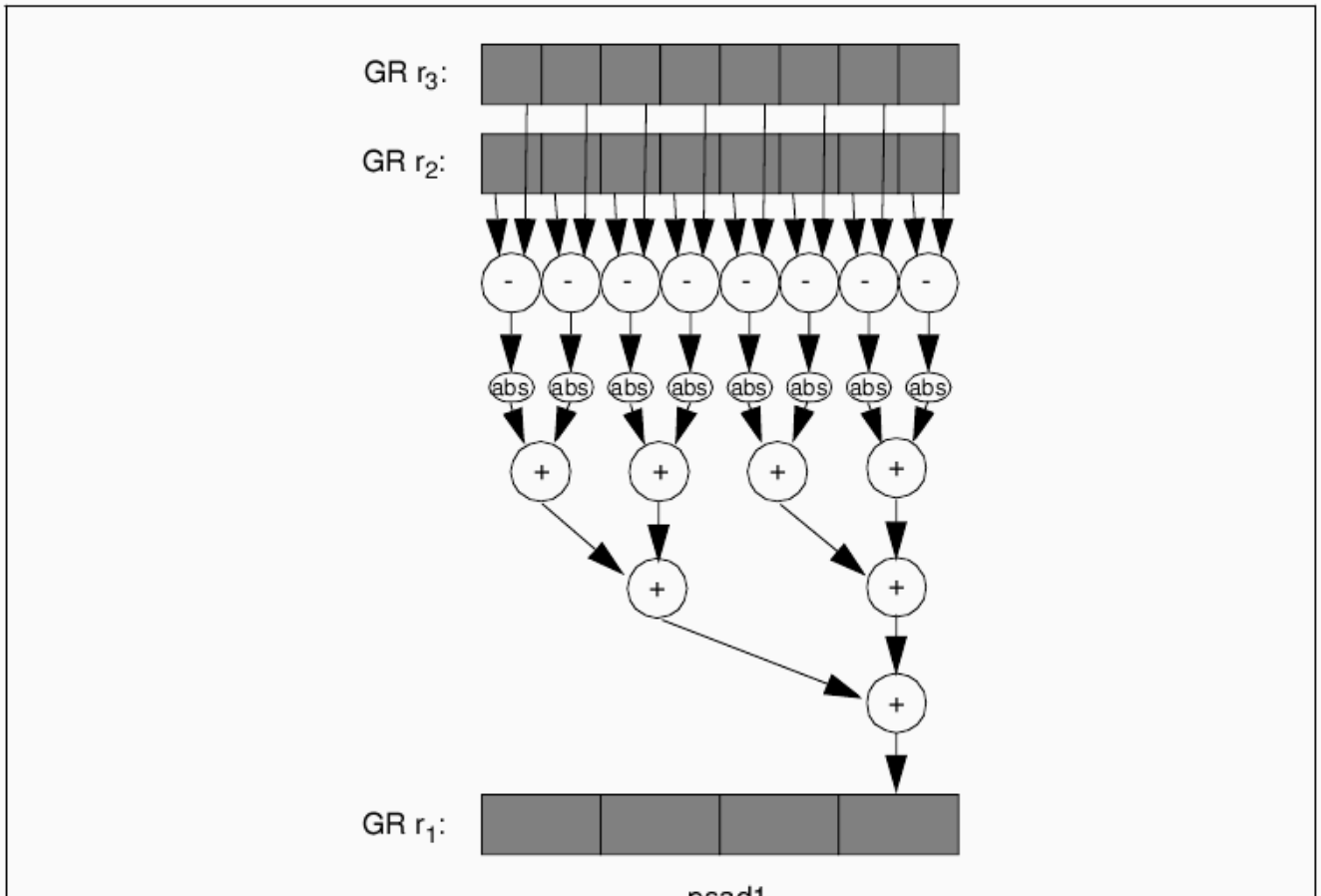
Whats new in a processor

psad — Parallel Sum of Absolute Difference

Format: (*qp*) psad1 $r_1 = r_2, r_3$ I2

Description: The unsigned 8-bit elements of GR r_2 are subtracted from the unsigned 8-bit elements of GR r_3. The absolute value of each difference is accumulated across the elements and placed in GR r_1.

Figure 2-38. Parallel Sum of Absolute Difference Example



How to make a compiler ?

1. Parse high level language to intermediate representation
2. Parse high level language to intermediate representation (again ?)
3. Parse high level language to intermediate representation (again ?)
4. Transform / optimize on IR
5. Generate the binary code

Intermediate representation (IR) are designed with an application in mind

But why a new compiler ?

- new architectures, new instruction set
- new applications (large, very large)
- data driven optimization needed
- but still the same languages

LLVM

- C, Objective C and C++ compiler, OpenCL, ...
- Clean documentation
- Clean infrastructure
- ... Clean licence
  `http://llvm.org/releases/2.6/LICENCE.TXT`
- Fast compiler, fast exec. code

LLVM goals

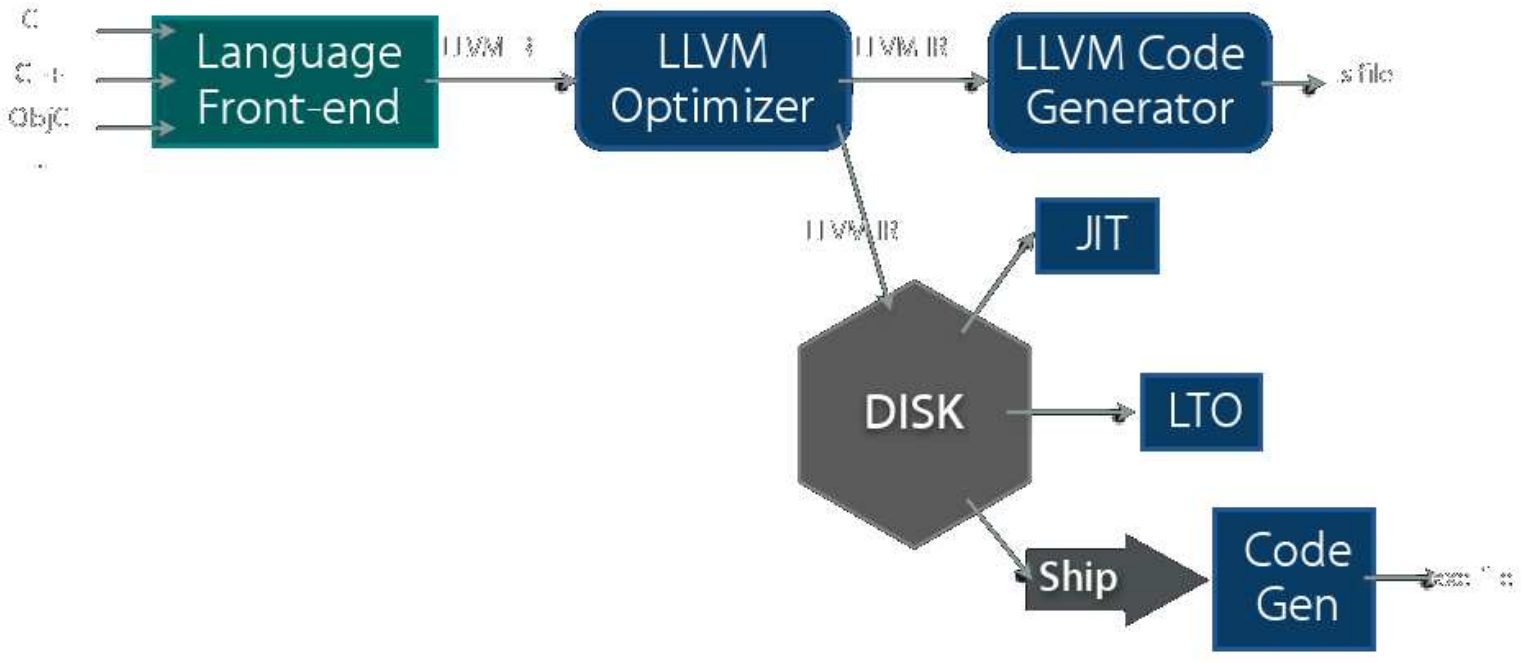
`http://llvm.org`

1. A compilation strategy designed to enable effective program optimization across the entire lifetime of a program.
2. A virtual instruction set
3. A compiler infrastructure
4. Low running cost (no garbage collection, runtime, etc)

Stats

```
cd /usr/ports/lang/clang
du -sh .; find . -type f |wc -l
 62M .
    9031
cd /usr/ports/lang/gcc45/
du -sh .; find . -type f |wc -l
329M .
   35018
```

Initial idea

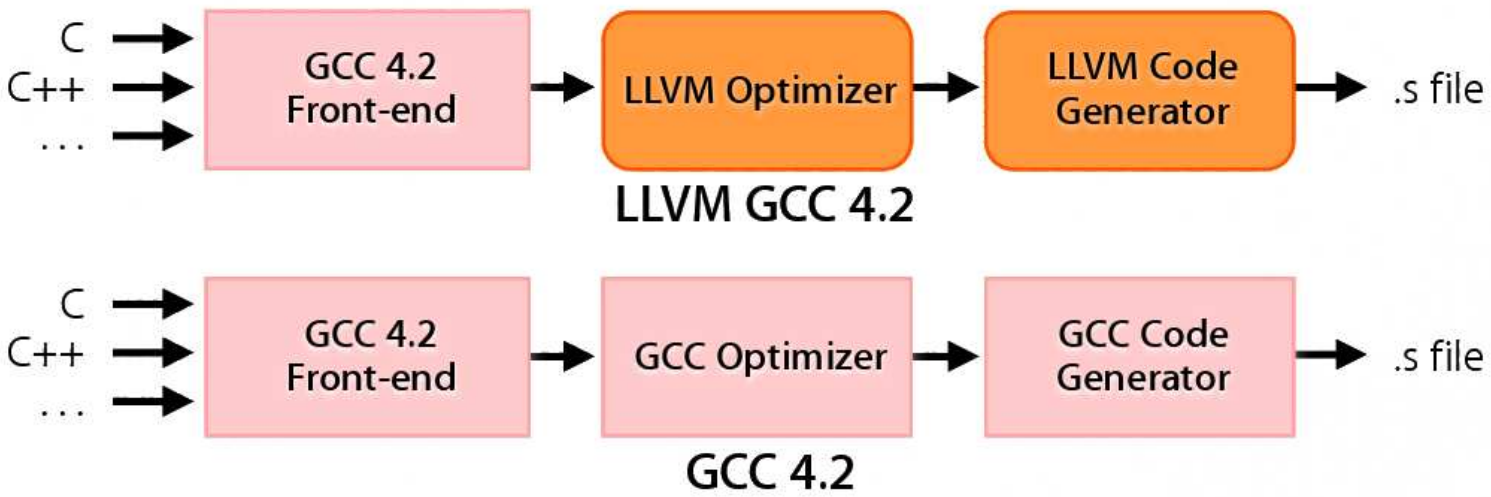


http://llvm.org/pubs/2002-12-LattnerMSThesis.html

http://www.llvm.org/pubs/2007-07-25-LLVM-2.0-and-Beyond.pdf

http://www.llvm.org/pubs/2008-05-17-BSDCan-LLVMIntro.pdf

LLVM-gcc initially



http://llvm.org/pubs/2008-10-04-ACAT-LLVM-Intro.pdf

OpenGL to LLVM

CLANG

- CLANG = new front end
- + LLVM optimizer
- + LLVM code gen.
- Apple sponsored
- C++ written

Jul 2007

All subsystems are documented

`http://llvm.org/docs/#subsystems`

- How to write a pass / a backend
- Code generator
- TableGen
- Alias analysis
- Source level debuging
- ... LTO

Rich low level format

`http://llvm.org/docs/LangRef.html`

Modular structure

Rich linkage type Optimisation interprocédurale (LTO)

Rich type system integer from 1 bit to 2^{23} bits length + vector data type

Rich constant description

Rich instruction set

Rich intrinsic functions

Usefull for what ?

- Dive into
- Specialized architecture (cell phones, e-reader, GPU, ...)
- Do something at run time

Results

- Boot FreeBSD
  `http://wiki.freebsd.org/BuildingFreeBSDWithClang`
- GPU programming (OpenCL)
- Cell phone (ARM based arch)