



Paul Osmialowski

# How The Flang Frontend Works

Introduction to the interior of the open-  
source Fortran frontend for LLVM

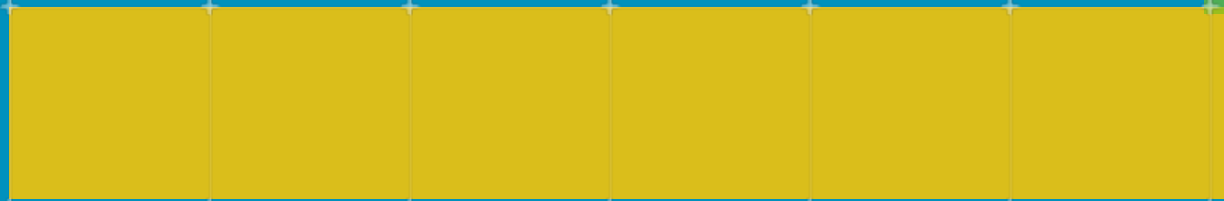
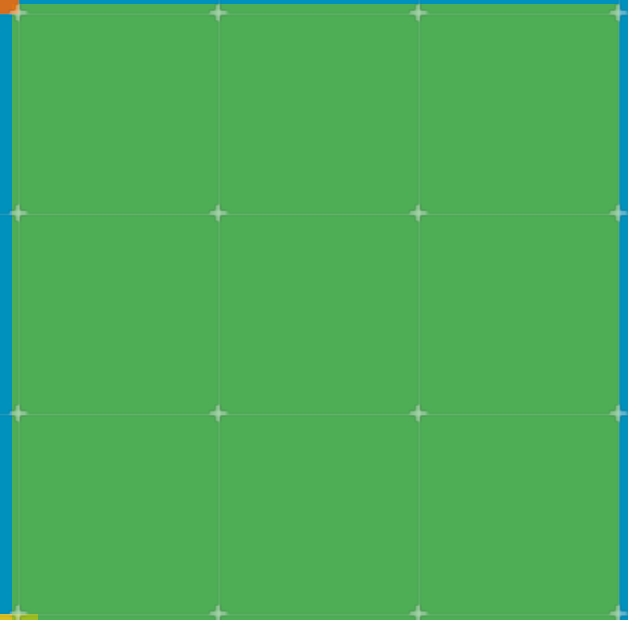
SC'17, LLVM-HPC2017 – Denver, Colorado

November 13<sup>th</sup>, 2017

# This talk...

- Why Flang?
- The Pipeline
- The Source Code
- Hello World
- Command Line Options
- Conclusions

# Why Flang?



# Flang – Fortran frontend for LLVM

In November 2015 LLNL announced an agreement with PGI® to create an open source Fortran compiler frontend for the LLVM compiler infrastructure

ARM agreed to work with PGI® on early previews to ensure architectural independence of generated LLVM IR **and the companion Fortran runtime library**



Lawrence Livermore National Laboratory

NOV. 13, 2015

```
PROGRAM MAIN
  PRINT *, 'ANNOUNCING...'
  PRINT *, 'OPEN SOURCE'
FORTRAN'
END
```

The Department of Energy's National Nuclear Security Administration and its three national labs will work with NVIDIA's PGI® software to create an open-source Fortran compiler designed for integration with the widely used LLVM compiler infrastructure. ([Download Image](#))

**NNSA, national labs team with Nvidia to develop open-source Fortran compiler technology**

# Horizon 2020, Mont Blanc 3

- H2020: biggest EU Research and Innovation programme with nearly €80 billion of funding available over 7 years (2014 to 2020).
- Mont Blanc 3: define the architecture of an Exascale-class compute node based on the ARM architecture, and capable of being manufactured at industrial scale.
  - Open-source Fortran compiler for ARMv8

# MONT-BLANC



# But wait... there is also llvm-flang on GitHub!

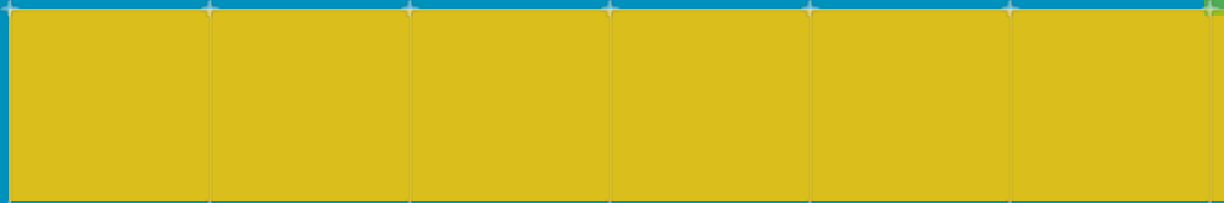
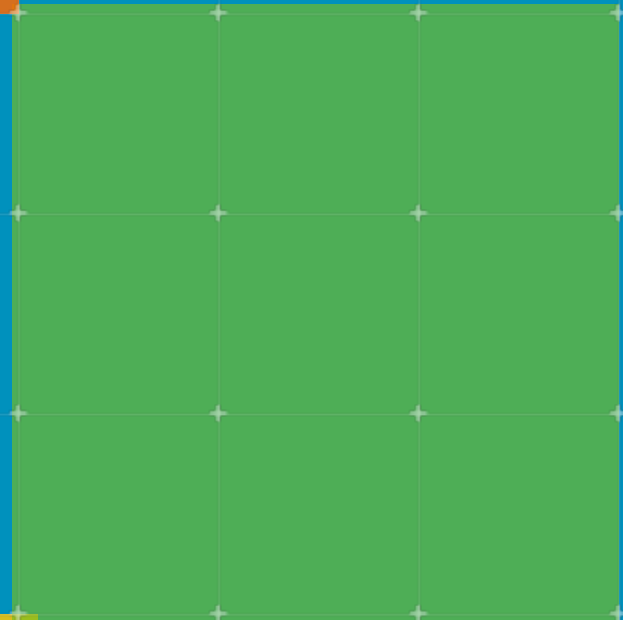
- Name reuse couldn't be avoided:
  - C/C++ -> Clang
  - Fortran -> Flang
- Forked from [CodethinkLabs/flang](https://github.com/CodethinkLabs/flang)
- From *README.txt*:

**Note:** This project is not related to the project of the same name based on PGI's Fortran compiler which can be found at <https://github.com/flang-compiler/flang>. See <https://github.com/llvm-flang/flang/issues/5> for details. (**Distinguish from PGI's Flang and forked sources**)

# So why .this. Flang?

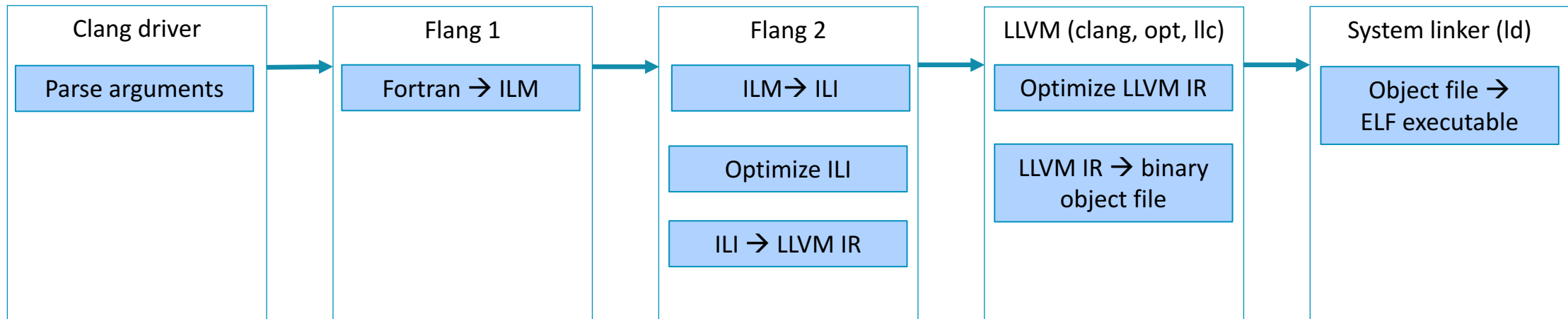
- May 2017: Public availability on GitHub
  - Steady stream of commits afterwards
- Good communication channels with original developers
  - GitHub bug tracker
  - *flang-compiler* Slack channel
- Many users do a lot of testing and reviewing
  - <http://thinkingeek.com/2017/06/17/walk-through-flang-part-1>
- Shares PGI's commercial compiler code, inherits a lot of its concepts and features
  - Two-pass LR(1) parser using parse tables generated by Parse Table Generator
  - Fortran 2003+ compatibility, <http://fortranwiki.org/fortran/show/Fortran+2003+status>
  - OpenMP v3+ compatibility
- Integral part of LLVM-based ARM HPC Compiler suite
- Available as a part of LLVM compiler family for OpenHPC project
  - Alternative to Intel and GCC compiler families

# The Pipeline





# How Flang fits into LLVM



# *flang1* and *flang2* phases

- Flang1 phases:
  - scanner, turns Fortran code into tokens.
  - parser, turns tokens into an AST and a symbol table.
  - transformer, turns the AST into a canonical AST.
  - output, turns the canonical AST into ILM (Intermediate Language Mnemonics).
- Flang2 phases:
  - expander, turns ILM into ILI (Intermediate Language Instructions).
  - optimizer, turns ILI into optimized ILI.
  - the bridge, turns optimized ILI into LLVM IR.

# Between flang1 and flang2: ILM text file

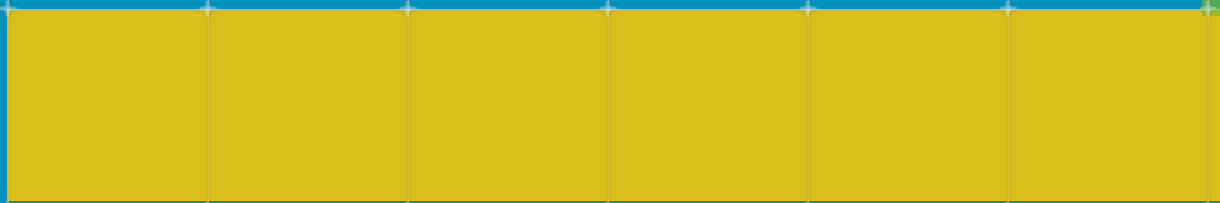
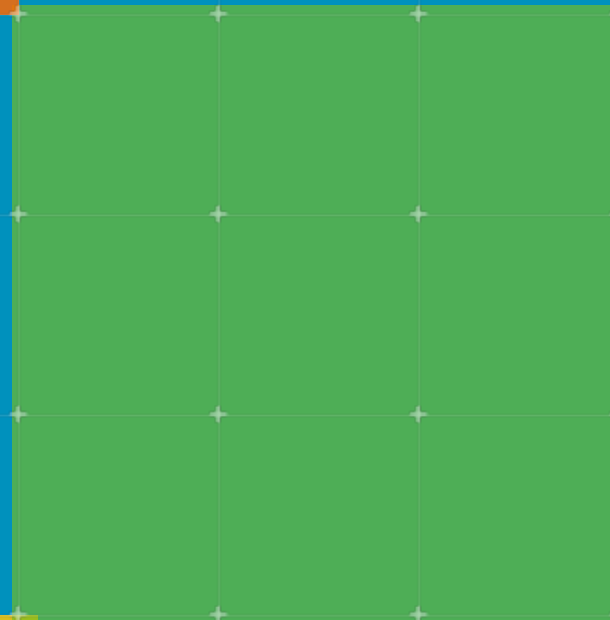
```
procedure:Program
d:6 I4
d:7 I8
d:8 R4
d:56 c 1:9
d:58 c 1:12
s:609 c n d:6 h- 0 0:
s:611 c n d:6 h- 1 0:
s:624 E E d:0 c+ a- a- a:0 C- d- d:0 c:0 e:3 i:0 l:1 m- p- r- r:0 p- p- s- d- c- n- c:0 r:0 p:0 a:0 v:0 i:0 i- c- d- a- v- 5:hello
s:626 c n d:56 h- 9:
=68656c6c6f2e663930
s:627 P E d:0 a:0 c+ d- d:0 d- c:0 f- i:0 m- n- p- r- p- p- c- s+ s- d- c- n- t+ r- r:0 C- u- i- r:0 p:0 v:0 i:0 i- c- m- c- i- c- f- a- v- p-
16:f90io_src_info03
s:628 c n d:6 h- 2 0:
s:629 c n d:6 h- 6 0:
s:630 P E d:6 a:0 c+ d+ d:0 d- c:0 f+ i:0 m- n- p- r- p- p- c- s+ s- d- c- n- t+ r- r:0 C- u- i- r:0 p:0 v:0 i:0 i- c- m- c- i- c- f- a- v- p-
16:f90io_print_init
s:631 V L d:6 a- a- a+ d+ d:0 m- c- c- h+ i- d- n- o- p- p- p:0 t- r- s- s+ t- p- u- i- p- t- e:0 t- t- v- a:0 c:0 c:0 l:1 m:0 p- e:0 p- p- C- a:0 c- d- p- s
- c- t- m- i- c- p:0 d:0 r- r- m- c- c- r- l- d- d- d:0 a- f- 5:z__io
s:633 c n d:58 h- 12:
=68656c6c6f20776f726c6420
s:634 c n d:8 h- 414570a4 0:
s:635 P E d:6 a:0 c+ d+ d:0 d- c:0 f+ i:0 m- n- p- r- p- p- c- s+ s- d- c- n- t+ r- r:0 C- u- i- r:0 p:0 v:0 i:0 i- c- m- c- i- c- f- a- v- p- 9:f90io_ldw
s:636 P E d:6 a:0 c+ d+ d:0 d- c:0 f+ i:0 m- n- p+ r- p- p- c- s- s- d- c- n- t- r- r:0 C- u- i- r:0 p:0 v:0 i:0 i- c- m- c- i- c- f- a- v- p-
15:f90io_sc_ch_ldw
s:637 c n d:6 h- e 0:
s:638 P E d:6 a:0 c+ d+ d:0 d- c:0 f+ i:0 m- n- p+ r- p- p- c- s- s- d- c- n- t- r- r:0 C- u- i- r:0 p:0 v:0 i:0 i- c- m- c- i- c- f- a- v- p-
14:f90io_sc_f_ldw
s:639 c n d:6 h- 1b 0:
s:640 P E d:6 a:0 c+ d+ d:0 d- c:0 f+ i:0 m- n- p- r- p- p- c- s+ s- d- c- n- t+ r- r:0 C- u- i- r:0 p:0 v:0 i:0 i- c- m- c- i- c- f- a- v- p-
13:f90io_ldw_end
s:643 L n d:0 c+ a- f- v- r:0 a:0 7:%I99999
s:645 c n d:7 h- 0 0 0:
s:646 c n d:6 h- 7fffffff 0:
s:647 P E d:0 a:0 c+ d- d:0 d- c:0 f- i:0 m- n- p- r- p- p- c- s+ s- d- c- n- t- r- r:0 C- u- i- r:0 p:0 v:0 i:0 i- c- m- c- i- c- f- a- v- p- 9:fort_init
end
```

- Not very human-readable
- ...but can be viewed in more readable form with use of debug command-line options described later

AST2ILM version 1/47

```
i0: BOS l1 n1 n0
i4: NOP
i5: -----
i0: BOS l1 n1 n0
i4: ICON s609
i6: CALL n1 s647 i4
i10: -----
i0: BOS l0 n1 n0
i4: ENLAB
i5: -----
i0: BOS l2 n1 n0
i4: FILE n2 n1 n1000
i8: ICON s628
i10: BASE s626
i12: FARG i8 t6
i15: FARG i10 t56
i18: UCALL n2 s627 i12 t6 i15 t56
i23: -----
i0: BOS l2 n1 n0
i4: FILE n2 n1 n2000
i8: BASE s631
i10: ICON s629
i12: ACON s645
i14: DPVAL i12
i16: ICON s609
i18: FARG i10 t6
i21: FARG i14 t6
i24: FARG i16 t6
i27: FARG i16 t6
i30: IUFUNC n4 s630 i18 t6 i21 t6 i24
t6 i27 t6
i37: IST i8 i30
i40: -----
```

# The Source Code



# Directory tree

- *lib, include* – auxiliary libraries shared by flang1, flang2 and the Fortran runtime library (hash tables, argument parsing, I/O utils).
- *runtime* – the Fortran runtime library.
- *tools/flang1* – flang1, Fortran to ILM translator.
- *tools/flang2* – flang2, ILM to LLVM IR translator.
- *tools/include* – contains header files with global definitions
- *utils* – the `errmsggen (errmsg.cpp)` utility, used to generate the error message definitions.

```
.
|-- docs
|-- include
|   |-- flang
|       |-- ADT
|       |-- ArgParser
|       |-- Error
|-- lib
|   |-- ADT
|   |-- ArgParser
|   |-- scutil
|-- runtime
|-- test
|-- tools
|   |-- flang1
|       |-- docs
|       |-- flang1exe
|       |-- include
|       |-- utils
|           |-- ast
|           |-- machar
|           |-- n2rst
|           |-- prstab
|           |-- symtab
|   |-- flang2
|       |-- docs
|       |-- flang2exe
|       |-- aarch64-Linux
|       |-- ppc64le-Linux
|       |-- x86_64-Linux
|       |-- include
|       |-- utils
|           |-- ilitp
|           |-- ilmtp
|           |-- machar
|           |-- n2rst
|           |-- symtab
|           |-- upper
|-- shared
-- utils
    -- errmsg
```

# tools/flang1

- *flang1exe* – the frontend, stage 1.
- *include/platform.h.in* – template for *platform.h* header file generated by CMake (platform specific definitions).
- *utils* – utilities required during flang1 building process:

- Header files generators
- Nroff-to-C converters
- Grammar definition and Parse Table generator

CMakeLists.txt	dtypeutil.c	inliner.c	parser.c	semfunc2.c
accpp.c	dtypeutil.h	inliner.h	pointsto.c	semgnr.c
assem.c	dummy.c	interf.c	psemant.c	semssp.c
ast.c	dump.c	interf.h	psemant2.c	semstk.h
astdf.c	exterf.c	invar.c	psemant3.c	semsym.c
astout.c	extern.h	iterat.c	psemantio.c	semtbp.c
bblock.c	feddesc.h	kwddf.h	psemssp.c	semutil.c
comm.c	fenddf.c	listing.c	pstride.c	semutil2.c
comm.h	fgraph.c	lower.c	redundss.c	soc.h
commdf.c	findloop.c	lower.h	rest.c	state.h
commgen.c	flgdf.h	lowerchk.c	rte.c	symacc.c
comminvar.c	flow.c	lowerexp.c	scan.c	symacc.h
commopt.c	fpp.c	lowerilm.c	scan.h	symtab.c
commopt.h	func.c	lowersym.c	scopestack.c	symutil.c
datadep.c	gbldefs.h	lz.c	semant.c	symutil.h
detect.c	global.h	main.c	semant.h	trace.h
dinit.c	hlvect.c	module.c	semant2.c	transfrm.c
dinit.h	hlvect.h	optdf.c	semant3.c	transfrm.h
dinitutil.c	hpfutl.c	optimize.c	semantio.c	version.c
dist.c	hpfutl.h	optimize.h	semast.c	version.h
dpm_out.c	induc.c	optutil.c	semfin.c	vsub.c
dpm_out.h	induc.h	outconv.c	semfunc.c	xref.c

# tools/flang2

- *flang2exe* – the frontend, stage 2.
- *include/platform.h.in* – template for *platform.h* header file generated by CMake (platform specific definitions).
- *utils* – utilities required during flang2 building process:
  - Nroff-to-C converters
    - ILM template utility; reads ILM definition file (*ilmtp.n*) and generates *ilmtp.h* and *ilmtpdf.h*.

CMakeLists.txt	expsmp.c	ll_ftn.c	regutil.c
aarch64-Linux	exputil.c	ll_structure.c	regutil.h
asm_anno.c	fastset.c	ll_structure.h	rsmove.c
asm_anno.h	fastset.h	ll_write.c	scope.c
assem.h	feddesc.h	ll_write.h	scope.h
bih.h	fenddf.c	llassem.c	semant.h
bihutil.c	gbldefs.h	llassem.h	semsym.c
cg.h	ili-rewrite.c	llassem_common.c	semutil0.c
cgllvm.h	ili-rewrite.h	lldebug.c	soc.h
cgmain.c	ili.h	lldebug.h	symacc.c
cgraph.h	ilidf.c	llopt.c	symacc.h
dinit.c	iliutil.c	llsched.c	syms.h
dinit.h	ilm.h	llutil.c	syntab.c
dinitutil.c	ilmutil.c	llutil.h	upper.c
dtypeutil.c	ilt.h	machreg.c	upper.h
dwarf2.h	iltutil.c	main.c	verify.c
dwarf_names.c	kmputcutil.c	mth.h	verify.h
exp_ftn.c	kmputcutil.h	mwd.c	version.c
exp_fvec.c	lili2llvm.c	mwd.h	version.h
exp_rte.c	lili2llvm.h	outliner.c	x86_64-Linux
expand.c	listing.c	outliner.h	xref.c
expatomic.c	ll_builder.c	ppc64le-Linux	
expdf.c	ll_builder.h	rbtree.c	
expreg.c	ll_dbgutl.c	rbtree.h	

# Documentation

- *nroff* format
  - various utilities (e.g. *groff* or *a2ps*) can be used to generate documentation pages from these source files.
- The same format is used for various definition files (*.n* files) from which e.g. certain C headers (*.h* files) are generated.
  - This ensures that all generated entities are also well documented.



# Interfacing LLVM

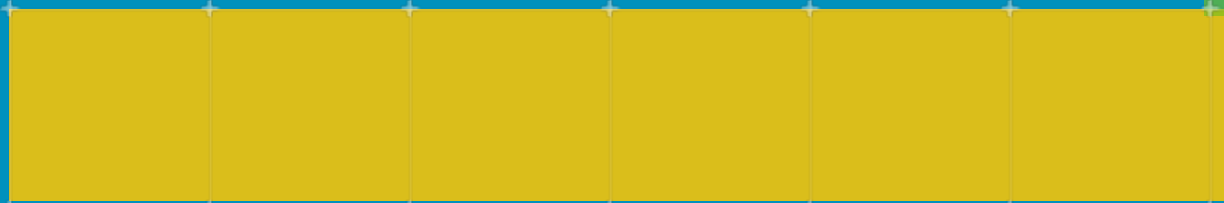
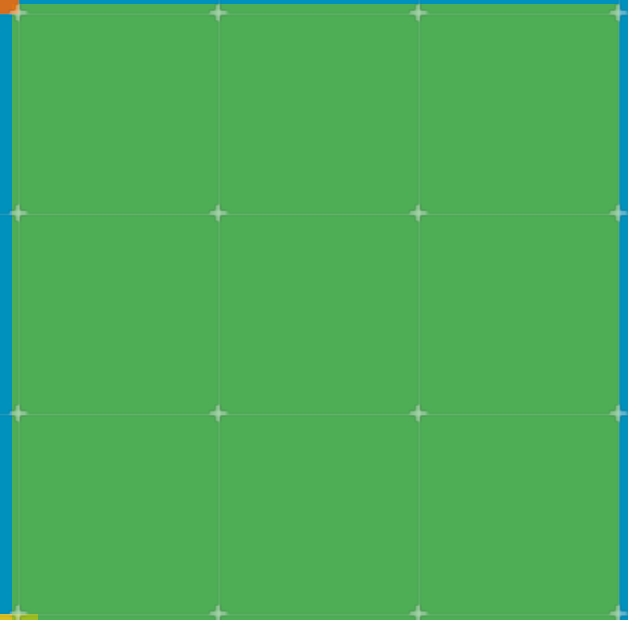
- LLVM API not used, LLVM IR textual representation generated through explicit operations on text strings
- Communication through temporary files, arranged by modified Clang compiler driver

# Multi-platform compatibility

- Currently AArch64, 64-bit Power and x86\_64
- *platform.h* file is generated by the CMake from *platform.h.in*
- The original Flang code evolved with strict ties to x86\_64 and recently added support for other architectures is implemented with **reuse of x86\_64 features**

```
/**** ARM - recycle FEATURE_ x64/x86 manifests ****/  
#if defined (TARGET_LLVM_ARM)  
#define FEATURE_SCALAR_NEON FEATURE_SCALAR_SSE  
#define FEATURE_NEON FEATURE_SSE  
#define FEATURE_FMA FEATURE_FMA3  
#endif
```

# Hello World



# Hello World – *hello.f90*

```
program hello
    print *, 'hello world', 12.34
end
```

# Compile *hello.f90* with Flang

```
$ flang -v -c hello.f90
clang version 4.0.1 (https://github.com/flang-compiler/clang.git
bcd9f99e52b47e13a64504a5783ce4eed40833835) (http://llvm.org/git/llvm.git
f3d3277bb713bb8aced9a7ac2e9b05c52d2844ee)
```

```
Target: aarch64-unknown-linux-gnu
Thread model: posix
Candidate multilib: .;@m64
Selected multilib: .;@m64
```

```
"flang1" hello.f90 -opt 0 -terse 1 -inform warn -nohpf -nostatic -y 129 2 -inform warn -x 19 0x400000 -quad -x 59 4 -x 15 2 -x 49 0x400004 -x 51 0x20
-x 57 0x4c -x 58 0x10000 -x 124 0x1000 -tp px -x 57 0xfb0000 -x 58 0x78031040 -x 47 0x08 -x 48 4608 -x 49 0x100 -stdinc $(dirname `which
flang`)/../include:/usr/local/include:$(dirname `which flang`)/../lib/clang/4.0.1/include:/usr/include/aarch64-linux-gnu:/include:/usr/include -def
unix -def __unix__ -def __linux__ -def linux -def __linux__ -def __linux__ -def __NO_MATH_INLINES -def __LP64__ -def __x86_64__ -def __x86_64__ -def
__LONG_MAX__=9223372036854775807L -def __SIZE_TYPE__=unsigned long int -def __PTRDIFF_TYPE__=long int -def __THROW= -def __extension__= -def
__amd64__ -def __amd64__ -def __k8__ -def __k8__ -def __PGLLVN__ -freeform -vect 48 -y 54 1 -x 70 0x40000000 -y 163 0xc0000000 -x 189 0x10 -stbfile
/tmp/hello-f2fb4a.stb -modexport /tmp/hello-f2fb4a.cmod -modindex /tmp/hello-f2fb4a.cmdx -output /tmp/hello-f2fb4a.ilm
```

```
"flang2" /tmp/hello-f2fb4a.ilm -ieee 1 -x 6 0x100 -x 42 0x400000 -y 129 4 -x 129 0x400 -fn hello.f90 -opt 0 -terse 1 -inform warn -y 129 2 -inform
warn -x 51 0x20 -x 119 0xa10000 -x 122 0x40 -x 123 0x1000 -x 127 4 -x 127 17 -x 19 0x400000 -x 28 0x40000 -x 120 0x10000000 -x 70 0x8000 -x 122 1 -x
125 0x20000 -quad -x 59 4 -tp px -x 120 0x1000 -x 124 0x1400 -y 15 2 -x 57 0x3b0000 -x 58 0x48000000 -x 49 0x100 -astype 0 -x 183 4 -x 121 0x800 -x
54 0x10 -x 70 0x40000000 -x 249 40 -x 124 1 -y 163 0xc0000000 -x 189 0x10 -y 189 0x40000000 -x 183 0x10 -stbfile /tmp/hello-f2fb4a.stb -asm
/tmp/hello-f2fb4a.ll
```

```
"clang-4.0" -cc1 -triple aarch64-unknown-linux-gnu -emit-obj -mrelax-all -disable-free -disable-llvm-verifier -discard-value-names -main-file-name
hello.f90 -mrelocation-model static -mthread-model posix -mdisable-fp-elim -fmath-errno -masm-verbose -mconstructor-aliases -fuse-init-array -target-
cpu generic -target-feature +neon -target-abi aapcs -v -dwarf-column-info -debugger-tuning=gdb -coverage-notes-file ./hello.gcno -resource-dir
$(dirname `which flang`)/../lib/clang/4.0.1 -fdebug-compilation-dir . -ferror-limit 19 -fmessage-length 316 -fallow-half-arguments-and-returns -fno-
signed-char -fobjc-runtime=gcc -fdiagnostics-show-option -fcolor-diagnostics -o hello.o -x ir /tmp/hello-f2fb4a.ll
```

# Flang1 tokens

tkntyp: PROGRAM tknval: 0 lineno: 1

tkntyp: <id name> tknval: 0 (hello) lineno: 1

tkntyp: END tknval: 0 lineno: 1

tkntyp: PRINT tknval: 0 lineno: 2

tkntyp: \* tknval: 0 lineno: 2

tkntyp: , tknval: 0 lineno: 2

tkntyp: <quoted string> tknval: 626 lineno: 2

tkntyp: , tknval: 0 lineno: 2

tkntyp: <real> tknval: 1095069860 lineno: 2

tkntyp: END tknval: 0 lineno: 2

tkntyp: <END stmt> tknval: 0 lineno: 3

tkntyp: END tknval: 0 lineno: 3

# Hello World

program hello

```
print *, 'hello world ', 12.34
```

end program hello

```
Subroutine: f90io_src_info03
Function: f90io_print_init
Function: f90io_sc_ch_ldw
Function: f90io_sc_f_ldw
Function: f90io_ldw_end
```

## AST

```
constant  hshlk/std:  0 type:character*12 opt=(0,0)
aptr:  23  sptr:  633 ("hello world ")

constant  hshlk/std:  0 type:integer opt=(0,0)
aptr:  25  sptr:  637 (14)

unaryop   hshlk/std:  0 type:integer alias:  0 callfg:0 opt=(0,0)
aptr:  26  lop :  25  optype:28

func-call hshlk/std:  0 type: alias:  0 callfg:1 opt=(0,0)
aptr:  27  lop:  28  argcnt:  2  args:  9
      (  0):  23
      (  1):  26

ident     hshlk/std:  0 type:integer alias:  0 callfg:0 opt=(0,0)
aptr:  28  sptr:  636 (f90io_sc_ch_ldw)
```

## ILM

```
----- lineno: 2 ----- global ILM index 0:0
 0 BOS      2      1      30
 4 FILE     2      1     3000
 8 BASE     303           ;z_io
10 BASE     304           ;"hello world "
12 ICON     308           ;14
14 DPVAL    12^
16 FARG     10^      56
19 FARG     14^      6
22 IFUNC    2      307   16^   19^ ;f90io_sc_ch_ldw
27 IST      8^      22^
```

## LLVM IR

```
{
...
store i32 %9, i32* %z_io_303, align 4, !dbg !19
%10 = bitcast [12 x i8]* @.C304_MAIN_ to i8*, !dbg !19
%11 = bitcast i32 (*)* @f90io_sc_ch_ldw to i32 (i8*, i32, i64)*, !dbg !19
%12 = call i32 @11 (i8* %10, i32 14, i64 12), !dbg !19
...
}
@.C304_MAIN_ = internal constant [12 x i8] [i8 104,i8 101,i8 108,i8 108,
                                           i8 111,i8 32,i8 119,i8 111,
                                           i8 114,i8 108,i8 100,i8 32]
```

## ILI

```
35 ACON      322~<z_io,0>
36 ST        34^   35^   3~ <z_io>   wd
37 ICON      323~<12>
38 ACON      324~<"hello world ",0>
39 ICON      308~<14>
40 KCON      325~<12>
41 ARGKR     40^   1^
42 ARGIR     39^   41^
43 ARGAR     38^   42^   0
44 GARG      40^   1^   7   1~ <?vol>
45 GARG      39^   44^   6   1~ <?vol>
46 GARG      38^   45^   56  0~ <?>
47 JSR       307~<f90io_sc_ch_ldw> 43^   48^-alt
48 GJSR      307~<f90io_sc_ch_ldw> 46^
49 DFRIR     47^   ir( 1)
```

The background features a blue grid with small white stars at the intersections. Three colored rectangles are positioned around the text: an orange rectangle at the top center, a green rectangle on the right side, and a yellow rectangle at the bottom left.

# Command Line Options



# Flang1, Flang2 command line options

- No '--help'!
  - ...but can be figured out from the source code.
- Intended for compiler driver control over the frontend
- More detailed control, -x and -y:
  - -x <number> <value>, e.g. -x 183 0x200000
  - Depending on the *number*, the *value* can be a plain integer value or a bit vector
  - They can be passed from flang to flang1 and/or flang2
    - To flang1: -Hx,124,0x10
    - To flang2: -Mx,183,0x200000
  - Multiple of flags like above can be given on flang invocation

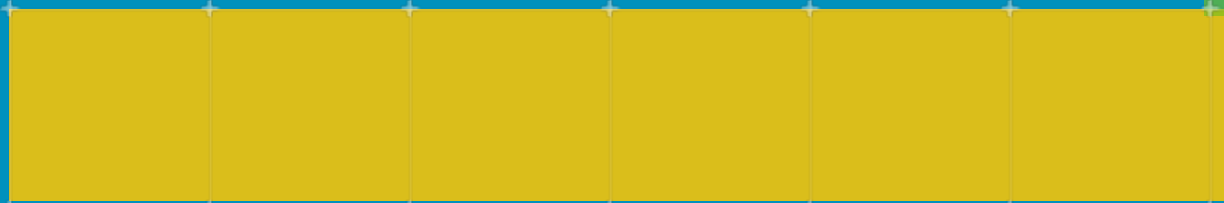
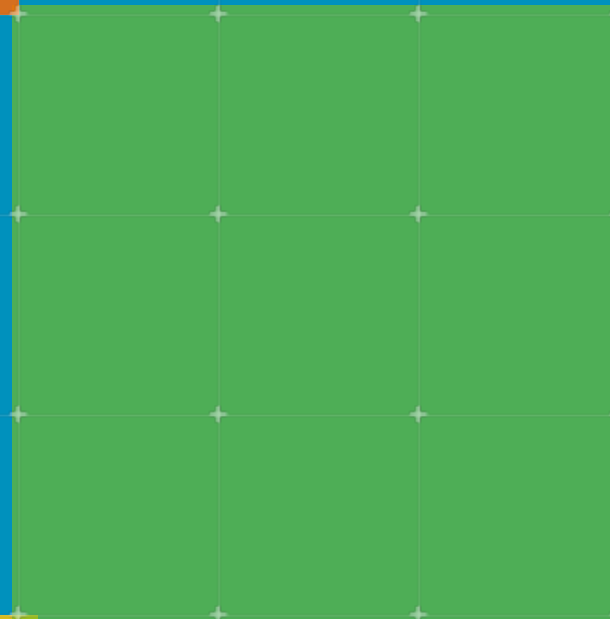
```
#define XBIT(n, m) (flg.x[n] & m)
```

```
if (XBIT(124, 0x10)) {  
    dt_int = DT_INT8; /* -i8 */  
} else {  
    dt_int = DT_INT;  
}
```

# Sneaking into internal data structures

- `-q` and `-qq` options
  - `-q <number> <value>`, e.g. `-q 0 1`
  - `-qq <phase name> <object name>`, e.g. `-qq parser ast`
    - List of all phase names and object names need to be found in source code (*dump\_map*)
- Similarly to `-x` and `-y`, they can be passed from `flang` to `flang1/flang2`, e.g.:
  - `-Hq,0,1 -Hqq,parser,symtab`
  - `-Mq,0,1 -Mq,10,2`

# Conclusions



# Pros and Cons

- Pros:
  - Based on mature code base
  - Under active development
  - Confirmed support for Fortran standards (including OpenMP pragmas)
  - Easily portable across 64-bit architectures
- Cons:
  - Based on mature code base
    - Mostly C
    - Coding style hugely different than LLVM's
    - Steep learning curve
  - Communication through temporary files only (it is slow!)

Thank You!

Danke!

Merci!

谢谢!

ありがとう!

Gracias!

Kiitos!

**arm**

The Arm trademarks featured in this presentation are registered trademarks or trademarks of Arm Limited (or its subsidiaries) in the US and/or elsewhere. All rights reserved. All other marks featured may be trademarks of their respective owners.



[www.arm.com/company/policies/trademarks](http://www.arm.com/company/policies/trademarks)