UFMG

Running the LLVM Tools

Compilers Laboratory

Fernando Magno Quintão Pereira
fernando@dcc.ufmg.br

Using LLVM

Using LLVM

DCC

Development Framework

Using LLVM

DCC

Development Framework

Standalone Tools

Tools

Tools

```
FileCheck            lli                  llvm-install-name-tool            llvm-rc
apinotes-test        lli-child-target     llvm-isel-fuzzer                  llvm-readelf
arcmt-test           llvm-PerfectShuffle  llvm-itanium-demangle-fuzzer      llvm-readobj
bugpoint             llvm-addr2line       llvm-jitlink                      llvm-reduce
c-arcmt-test         llvm-ar              llvm-jitlink-executor             llvm-rtdyld
c-index-test         llvm-as              llvm-lib                          llvm-size
clang                llvm-bcanalyzer      llvm-libtool-darwin               llvm-special-case-list-fuzzer
clang++              llvm-bitcode-stp     llvm-link                         llvm-split
clang-12             llvm-c-test          llvm-lipo                         llvm-stress
clang-check          llvm-cat             llvm-lit                          llvm-strings
clang-cl             llvm-cfi-verify      llvm-locstats                     llvm-strip
clang-cpp            llvm-config          llvm-lto                          llvm-symbolizer
clang-diff           llvm-cov             llvm-lto2                         llvm-tblgen
clang-extdef-map     llvm-cvtres          llvm-mc                           llvm-undname
clang-format         llvm-cxxdump         llvm-mca                          llvm-xray
clang-import-test    llvm-cxxfilt         llvm-ms-demangle-fuzzer           llvm-yaml-numeric-parser-fuzzer
clang-offld-bundler  llvm-cxxmap          llvm-ml                           not
clang-offld-wrapper  llvm-diff            llvm-modextract                   obj2yaml
clang-refactor       llvm-dis             llvm-mt                           opt
clang-rename         llvm-dlltool         llvm-nm                           sancov
clang-scan-deps      llvm-dwarfdump       llvm-objcopy                      sanstats
clang-tblgen         llvm-dwp             llvm-objdump                      scan-build
count                llvm-elfabi          llvm-opt-fuzzer                   scan-view
diagtool             llvm-exegesis        llvm-opt-report                   split-file
dsymutil             llvm-extract         llvm-pdbutil                      verify-uselistorder
hmaptool             llvm-gsymutil        llvm-profdata                     yaml-bench
llc                  llvm-ifs             llvm-ranlib                       yaml2obj
```

Tools

DCC

```
FileCheck
apinotes-test
arcmt-test
bugpoint
c-arcmt-test
c-index-test
clang
clang++
clang-12
clang-check
clang-cl
clang-cpp
clang-diff
clang-extdef-map
clang-format
clang-import-test
clang-offld-bundler
clang-offld-wrapper
clang-refactor
clang-rename
clang-scan-deps
clang-tblgen
count
diagtool
dsymutil
hmaptool
llc
lli
lli-child-target
llvm-PerfectShuffle
llvm-addr2line
llvm-ar
llvm-as
llvm-bcanalyzer
llvm-bitcode-stp
llvm-c-test
llvm-cat
llvm-cfi-verify
llvm-config
llvm-cov
llvm-cvtres
llvm-cxxdump
llvm-cxxfilt
llvm-cxxmap
llvm-diff
llvm-dis
llvm-dlltool
llvm-dwarfdump
llvm-dwp
llvm-elfabi
llvm-exegesis
llvm-extract
llvm-gsymutil
llvm-ifs
llvm-install-name-tool
llvm-isel-fuzzer
llvm-itanium-demangle-fuzzer
llvm-jitlink
llvm-jitlink-executor
llvm-lib
llvm-libtool-darwin
llvm-link
llvm-lipo
llvm-lit
llvm-locstats
llvm-lto
llvm-lto2
llvm-mc
llvm-mca
llvm-ms-demangle-fuzzer
llvm-ml
llvm-modextract
llvm-mt
llvm-nm
llvm-objcopy
llvm-objdump
llvm-opt-fuzzer
llvm-opt-report
llvm-pdbutil
llvm-profdata
llvm-ranlib
llvm-rc
llvm-readelf
llvm-readobj
llvm-reduce
llvm-rtdyld
llvm-size
llvm-special-case-list-fuzzer
llvm-split
llvm-stress
llvm-strings
llvm-strip
llvm-symbolizer
llvm-tblgen
llvm-undname
llvm-xray
llvm-yaml-numeric-parser-fuzzer
not
obj2yaml
opt
sancov
sanstats
scan-build
scan-view
split-file
verify-uselistorder
yaml-bench
yaml2obj
```

Tools

DCC

FileCheck
apinotes-test
arcmt-test
bugpoint
c-arcmt-test
c-index-test
clang
clang++
clang-12
clang-check
clang-cl
clang-cpp
clang-diff
clang-extdef-map
clang-format
clang-import-test
clang-offld-bundler
clang-offld-wrapper
clang-refactor
clang-rename
clang-scan-deps
clang-tblgen
count
diagtool
dsymutil
hmaptool
llc

lli
lli-child-target
llvm-PerfectShuffle
llvm-addr2line
llvm-ar
llvm-as
llvm-bcan
llvm-bitc
llvm-c-te
llvm-cat
llvm-cfi-
llvm-conf
llvm-cov
llvm-cvtr
llvm-cxxd
llvm-cxxf
llvm-cxxm
llvm-diff
llvm-dis
llvm-dlltool
llvm-dwarfdump
llvm-dwp
llvm-elfabi
llvm-exegesis
llvm-extract
llvm-gsymutil
llvm-ifs

llvm-install-name-tool
llvm-isel-fuzzer
llvm-itanium-demangle-fuzzer
llvm-jitlink
utor
in
fuzzer
llvm-modextract
llvm-mt
llvm-nm
llvm-objcopy
llvm-objdump
llvm-opt-fuzzer
llvm-opt-report
llvm-pdbutil
llvm-profdata
llvm-ranlib

llvm-rc
llvm-readelf
llvm-readobj
llvm-reduce
llvm-rtdyld
llvm-size
llvm-special-case-list-fuzzer
llvm-split
llvm-stress
llvm-strings
llvm-strip
llvm-symbolizer
llvm-tblgen
llvm-undname
llvm-xray
llvm-yaml-numeric-parser-fuzzer
not
obj2yaml
opt
sancov
sanstats
scan-build
scan-view
split-file
verify-uselistorder
yaml-bench
yaml2obj

clang

opt

llc

Compiler Organization

Compiler Organization

DCC

Front end

Middle end

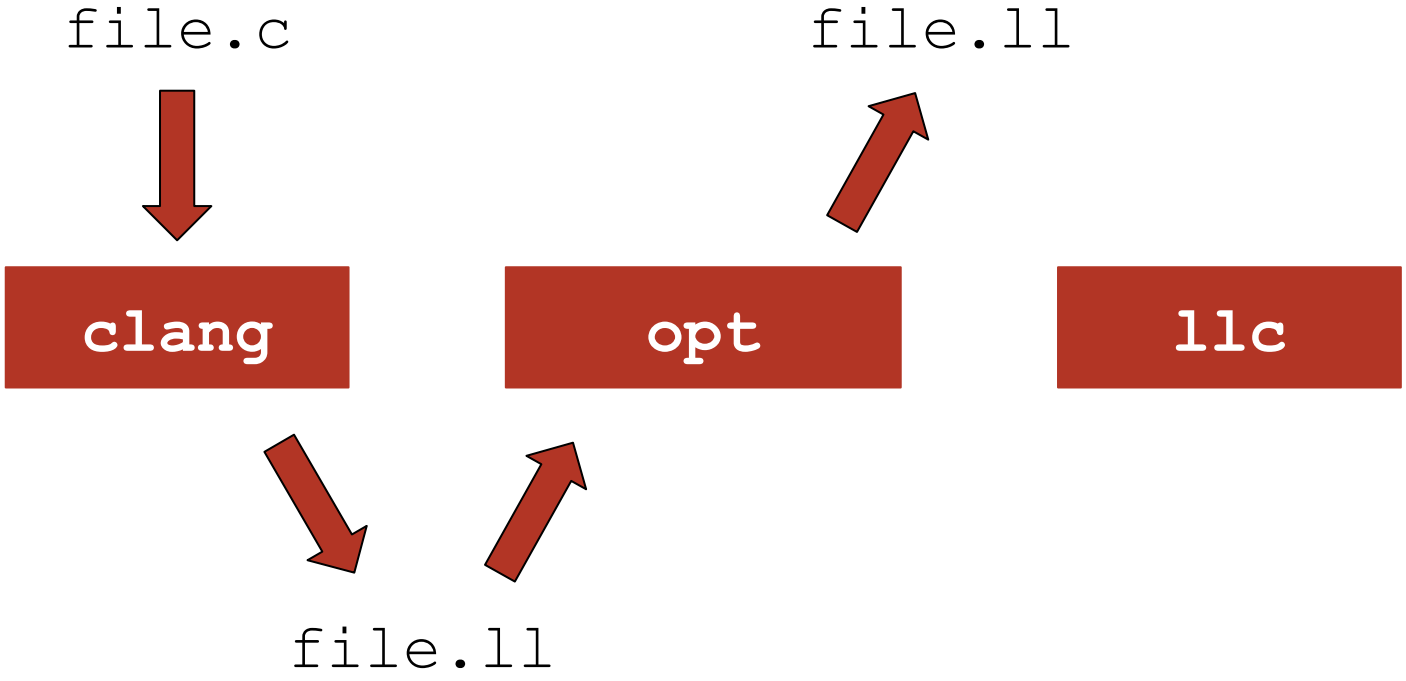
Back end

Compiler Organization

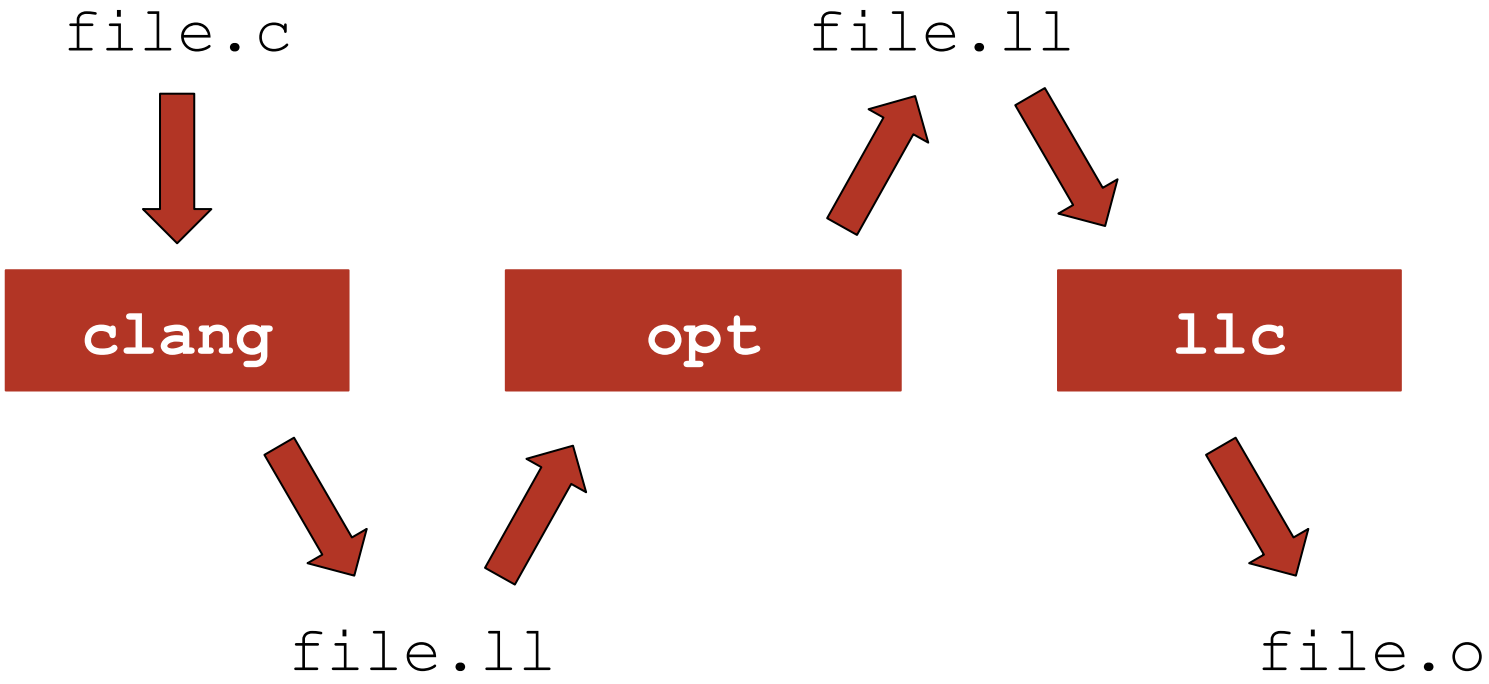
DCC

Compiler Organization

Compiler Organization



Compiler Organization



clang

DCC

clang

DCC

```
void prefix_sum(int *src, int *dst, int N) {
  if (0 < N) {
    int i = 0;
    do {
      int tmp = 0;
      int j = 0;
      if (j < i) {
        do {
          tmp += src[j];
          j++;
        } while (j < i);
        dst[i] = tmp;
      }
      i++;
    } while (i < N);
  }
}
```

clang

DCC

```
void prefix_sum(int *src, int *dst, int N) {
  if (0 < N) {
    int i = 0;
    do {
      int tmp = 0;
      int j = 0;
      if (j < i) {
        do {
          tmp += src[j];
          j++;
        } while (j < i);
        dst[i] = tmp;
      }
      i++;
    } while (i < N);
  }
}
```

```
$ clang -S file.c -o file.ll
```

clang

DCC

```
void prefix_sum(int *src, int *dst, int N) {
  if (0 < N) {
    int i = 0;
    do {
      int tmp = 0;
      int j = 0;
      if (j < i) {
        do {
          tmp += src[j];
          j++;
        } while (j < i);
        dst[i] = tmp;
      }
      i++;
    } while (i < N);
  }
}
```

```
$ clang -S file.c -o file.ll
```

clang

```
void prefix_sum(int *src, int *dst, int N) {
  if (0 < N) {
    int i = 0;
    do {
      int tmp = 0;
      int j = 0;
      if (j < i) {
        do {
          tmp += src[j];
          j++;
        } while (j < i);
        dst[i] = tmp;
      }
      i++;
    } while (i < N);
  }
}
```

```
$ clang -S file.c -o file.ll
```

clang

DCC

```
void prefix_sum(int *src, int *dst, int N) {
  if (0 < N) {
    int i = 0;
    do {
      int tmp = 0;
      int j = 0;
      if (j < i) {
        do {
          tmp += src[j];
          j++;
        } while (j < i);
        dst[i] = tmp;
      }
      i++;
    } while (i < N);
  }
}
```

```
$ clang -S file.c -o file.ll
```

clang

DCC

```
$ clang -S file.c -o file.ll
$ cat file.ll
```

clang

```
.text
.file   "file.c"
.globl  prefix_sum
.p2align        4, 0x90
.type
prefix_sum,@function
prefix_sum:
  .cfi_startproc
# %bb.0:
  pushq %rbp
  .cfi_def_cfa_offset 16
  .cfi_offset %rbp, -16
  movq  %rsp, %rbp
  .cfi_def_cfa_register
%rbp
  xorl  %eax, %eax
  movq  %rdi, -8(%rbp)
  movq  %rsi, -16(%rbp)
  movl  %edx, -20(%rbp)
  cmpl  -20(%rbp), %eax
  jge   .LBB0_10
# %bb.1:
  movl  $0, -24(%rbp)
  ...
```

```
$ clang -S file.c -o file.ll
$ cat file.ll
```

clang

DCC

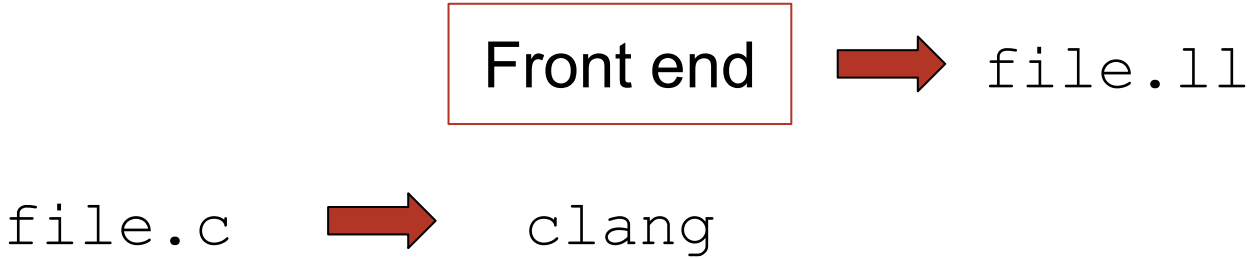
```
$ clang -S -emit-llvm file.c -o file.ll
```

clang

```
$ clang -S -emit-llvm file.c -o file.ll
$ cat file.ll
```

```
define dso_local
void @prefix_sum(i32* %src, i32* %dst, i32 %N)
#0 {
entry:
  %src.addr = alloca i32*, align 8
  %dst.addr = alloca i32*, align 8
  %N.addr = alloca i32, align 4
  %i = alloca i32, align 4
  %tmp = alloca i32, align 4
  ...
```

Other Front Ends



Other Front Ends

DCC

Front end → file.ll

file.c → clang

file.rs → rustc

Other Front Ends

DCC

Front end → `file.ll`

`file.c` → `clang`

`file.rs` → `rustc`

`file.jl` → `julia`

Other Front Ends

DCC

Front end → file.ll

file.c → clang

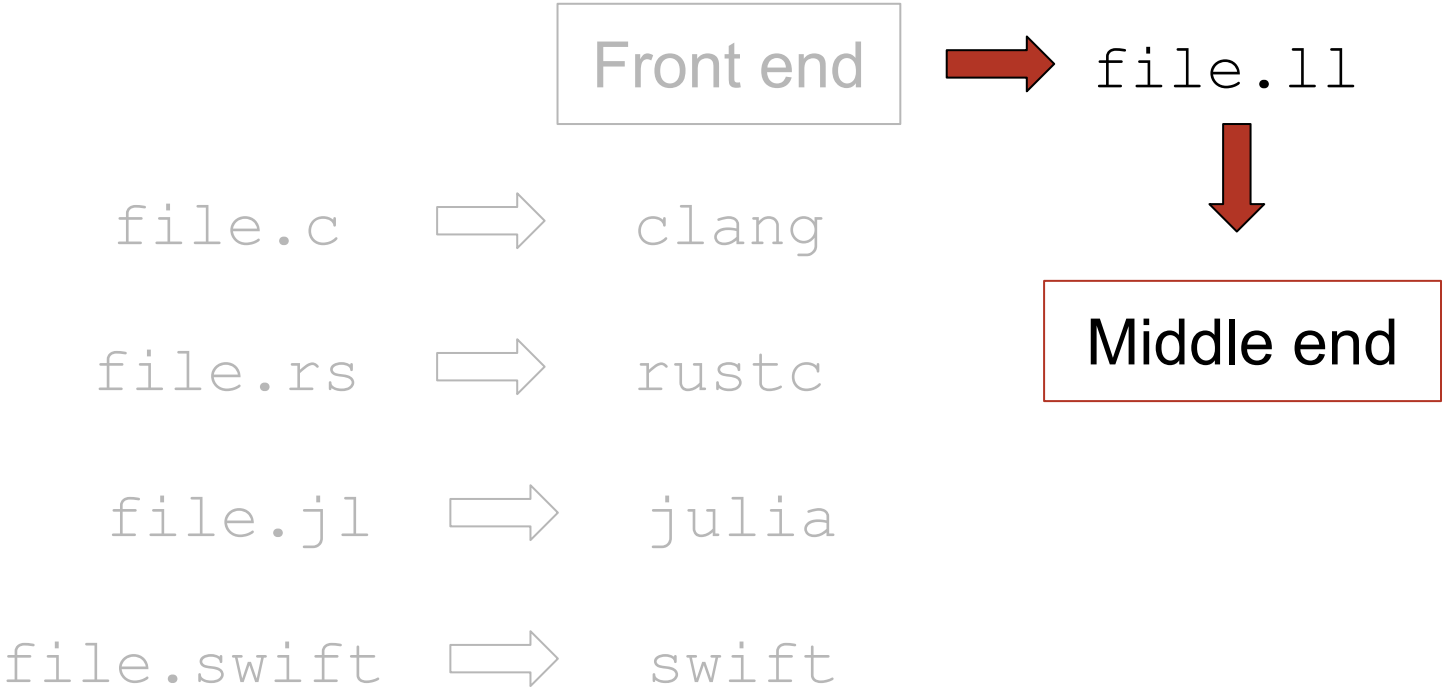
file.rs → rustc

file.jl → julia

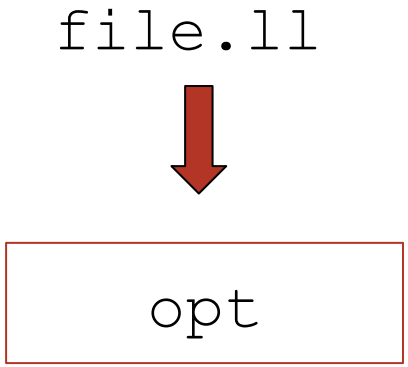
file.swift → swift

The Middle End

DCC

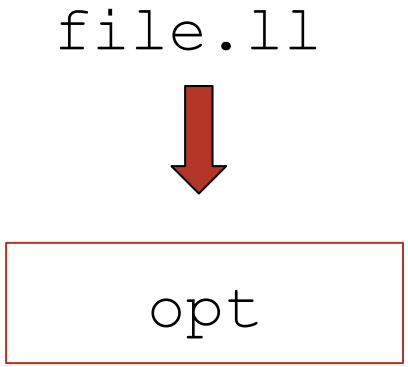


opt



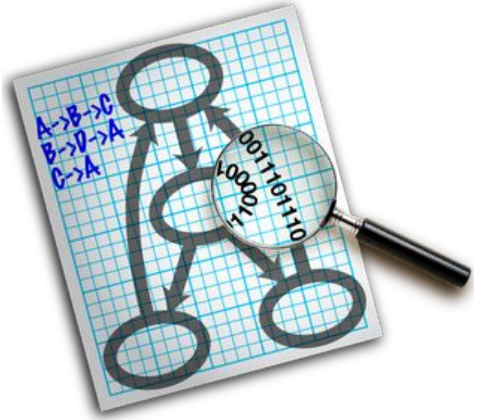
opt

DCC




```
$ opt -dot-cfg file.ll -disable-output
```

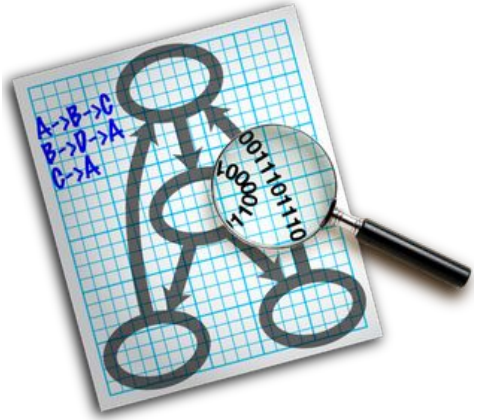
DOT



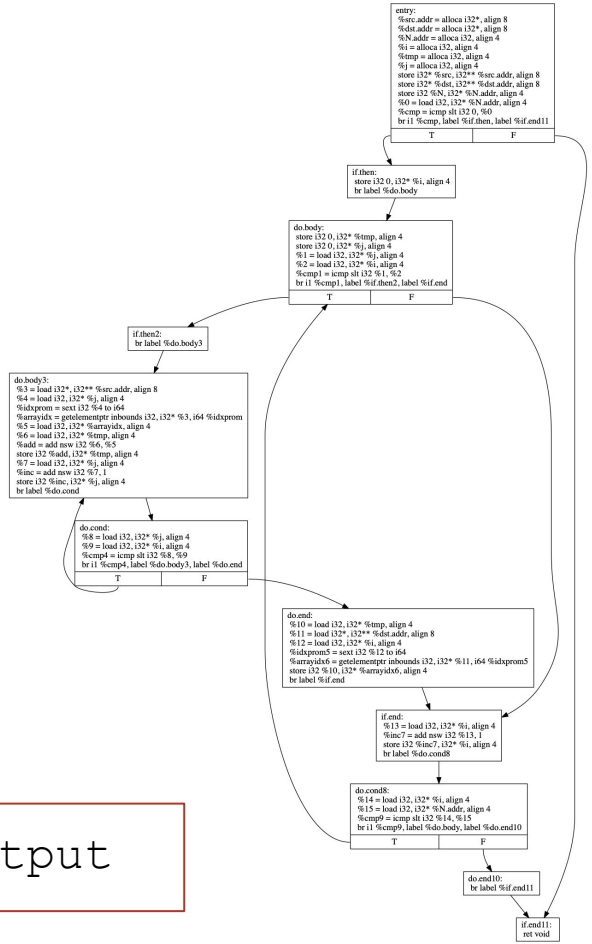
graphviz

```
$ opt -dot-cfg file.ll -disable-output
```

DOT



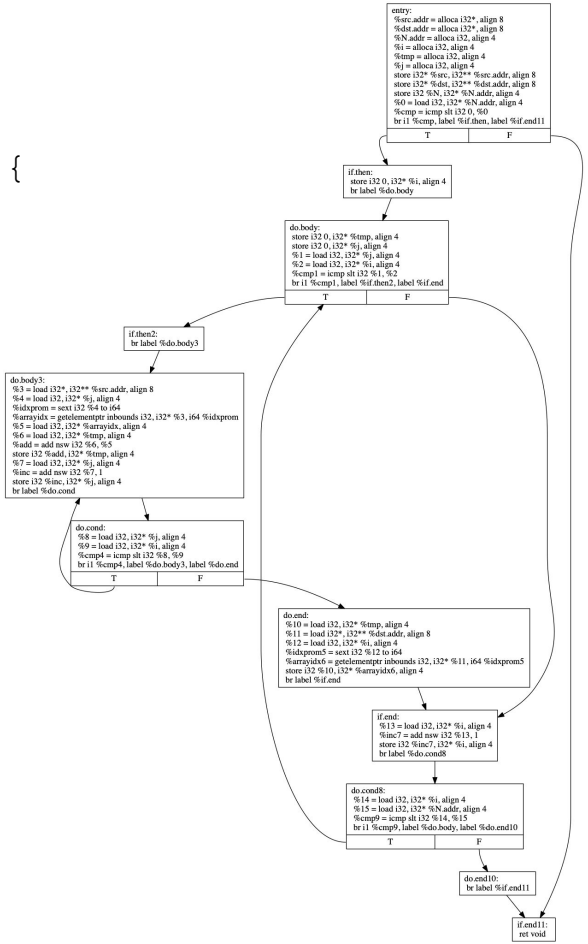
graphviz




```
$ opt -dot-cfg file.ll -disable-output
```

DOT

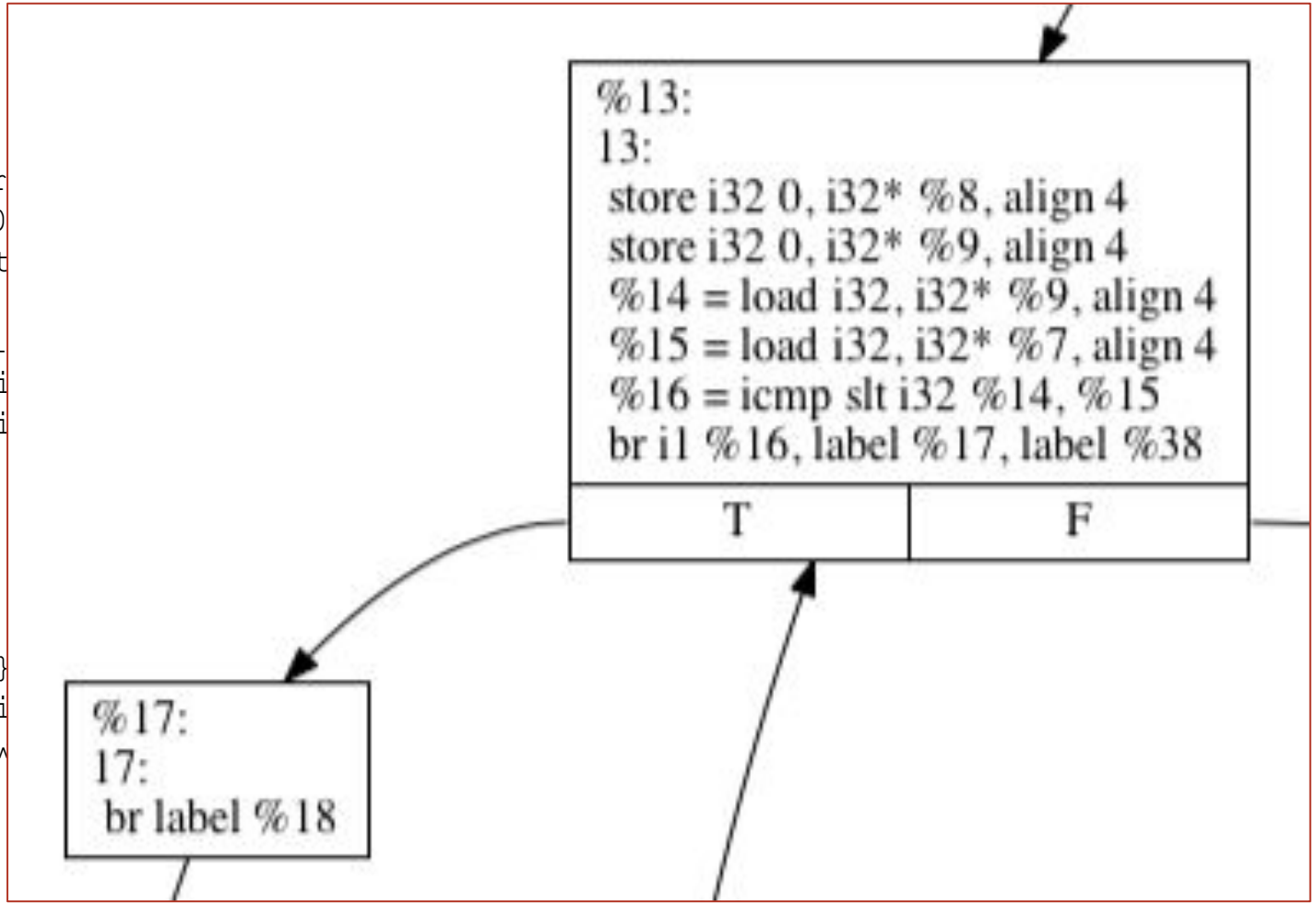
```
void prefix_sum(int *src, int *dst, int N) {
  if (0 < N) {
    int i = 0;
    do {
      int tmp = 0;
      int j = 0;
      if (j < i) {
        do {
          tmp += src[j];
          j++;
        } while (j < i);
        dst[i] = tmp;
      }
      i++;
    } while (i < N);
  }
}
```



DOT

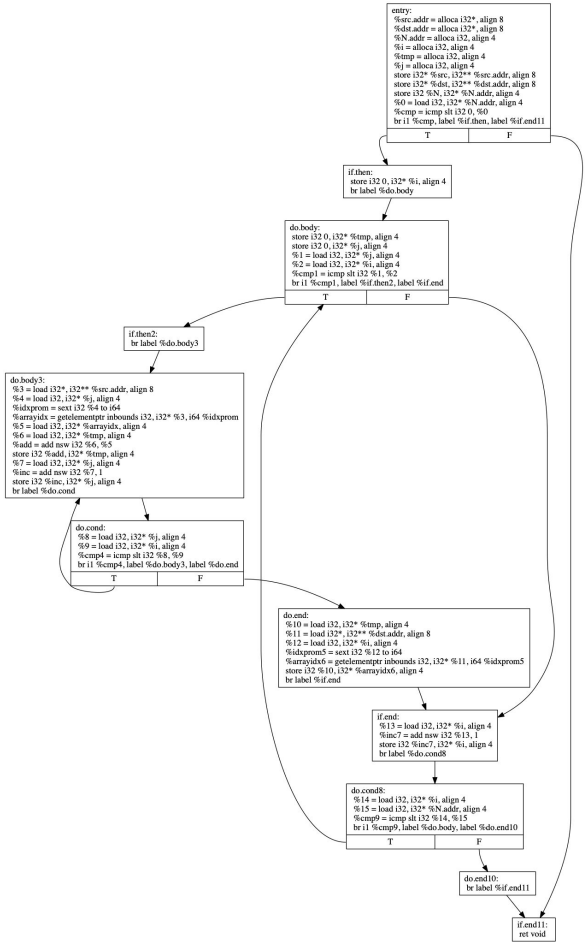
```
void pr
  if (0
     int
     do
       i
       i
       i

       }
       i
     } w
  }
}
```



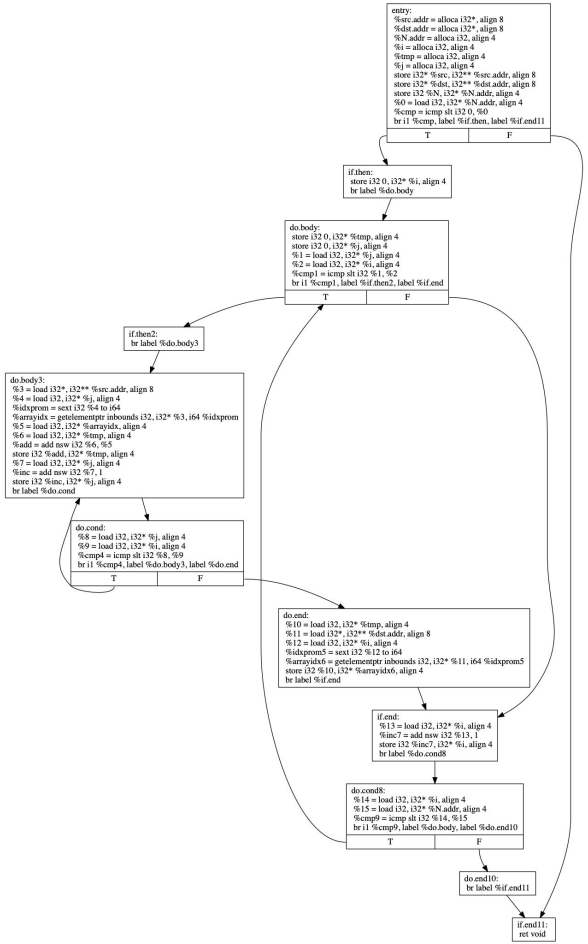
Back End

Back end



llc


llc


entry:
%src.addr = alloca i32*, align 8
%dst.addr = alloca i32*, align 8
%N.addr = alloca i32, align 4
%i = alloca i32, align 4
%tmp = alloca i32, align 4
%j = alloca i32, align 4
store i32* %src, i32** %src.addr, align 8
store i32* %dst, i32** %dst.addr, align 8
store i32 %N, i32* %N.addr, align 4
%0 = load i32, i32* %N.addr, align 4
%cmp = icmp slt i32 0, %0
br i1 %cmp, label %if.then, label %if.end11
T
F
if.then:
store i32 0, i32* %i, align 4
br label %do.body
do.body:
store i32 0, i32* %tmp, align 4
store i32 0, i32* %j, align 4
%1 = load i32, i32* %i, align 4
%2 = load i32, i32* %i, align 4
%cmp1 = icmp slt i32 %1, %2
br i1 %cmp1, label %if.then2, label %if.end
T
F
if.then2:
br label %do.body3
do.body3:
%3 = load i32*, i32** %src.addr, align 8
%4 = load i32, i32* %j, align 4
%idxprom = sext i32 %4 to i64
%arrayidx = getelementptr inbounds i32, i32* %3, i64 %idxprom
%5 = load i32, i32* %arrayidx, align 4
%6 = load i32, i32* %tmp, align 4
%add = add nsw i32 %6, %5
store i32 %add, i32* %tmp, align 4
%7 = load i32, i32* %j, align 4
%inc = add nsw i32 %7, 1
store i32 %inc, i32* %j, align 4
br label %do.cond
do.cond:
%8 = load i32, i32* %j, align 4
%9 = load i32, i32* %i, align 4
%cmp4 = icmp slt i32 %8, %9
br i1 %cmp4, label %do.body3, label %do.end
T
F
do.end:
%10 = load i32, i32* %tmp, align 4
%11 = load i32*, i32** %dst.addr, align 8
%12 = load i32, i32* %i, align 4
%idxprom5 = sext i32 %12 to i64
%arrayidx6 = getelementptr inbounds i32, i32* %11, i64 %idxprom5
store i32 %10, i32* %arrayidx6, align 4
br label %if.end
if.end:
%13 = load i32, i32* %i, align 4
%inc7 = add nsw i32 %13, 1
store i32 %inc7, i32* %i, align 4
br label %do.cond8
do.cond8:
%14 = load i32, i32* %i, align 4
%15 = load i32, i32* %N.addr, align 4
%cmp9 = icmp slt i32 %14, %15
br i1 %cmp9, label %do.body, label %do.end10
T
F
do.end10:
br label %if.end11
if.end11:
ret void

llc

arm

mips

ppc32

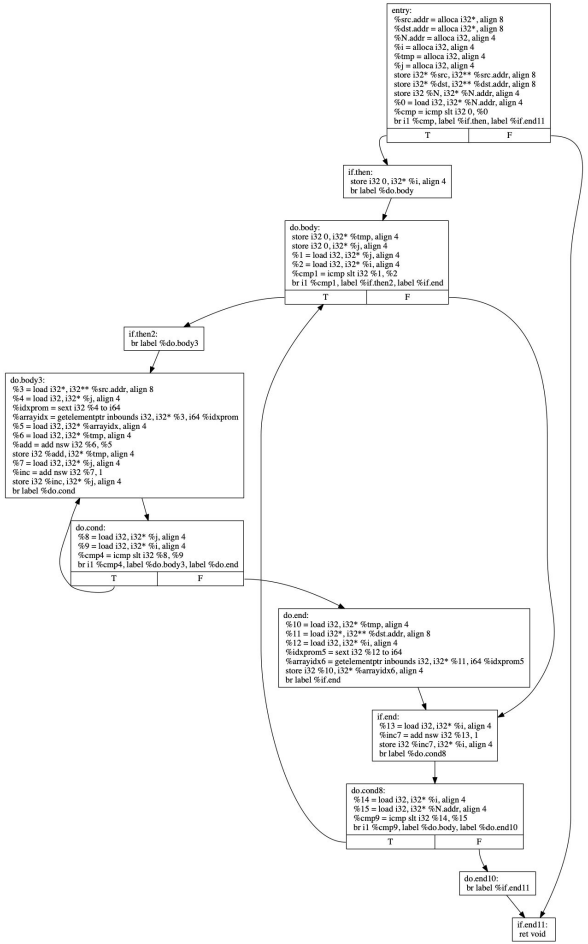
riscv32

sparc

x86

...

llc



llc

```
$ llc --version
```

llc

```
$ llc --version
```

```
Registered Targets:
  aarch64        - AArch64 (little endian)
  aarch64_32 - AArch64 (little endian ILP32)
  aarch64_be - AArch64 (big endian)
  amdgcn         - AMD GCN GPUs
  arm            - ARM
  arm64          - ARM64 (little endian)
  arm64_32   - ARM64 (little endian ILP32)
  armeb          - ARM (big endian)
  bpf            - BPF (host endian)
  bpfeb          - BPF (big endian)
  bpfel          - BPF (little endian)
  hexagon        - Hexagon
  lanai          - Lanai
  mips           - MIPS (32-bit big endian)
  mips64         - MIPS (64-bit big endian)
  mips64el   - MIPS (64-bit little endian)
  mipsel         - MIPS (32-bit little endian)
  msp430         - MSP430 [experimental]
  nvptx          - NVIDIA PTX 32-bit
  nvptx64        - NVIDIA PTX 64-bit
  ppc32          - PowerPC 32
  ppc64          - PowerPC 64
  ppc64le        - PowerPC 64 LE
  r600           - AMD GPUs HD2XXX-HD6XXX
  riscv32        - 32-bit RISC-V
  riscv64        - 64-bit RISC-V
  sparc          - Sparc
  sparcel        - Sparc LE
  sparcv9        - Sparc V9
  systemz        - SystemZ
  thumb          - Thumb
  thumbeb        - Thumb (big endian)
  wasm32         - WebAssembly 32-bit
  wasm64         - WebAssembly 64-bit
  x86            - 32-bit X86: Pentium-Pro and above
  x86-64         - 64-bit X86: EM64T and AMD64
  xcore          - XCore
```

llc

DCC

```
$ llc file.ll -march=x86 -o file.x86
```

llc

```
$ llc file.ll -march=x86 -o file.x86
$ cat file.x86
```

```
  .text
  .file "file.c"
  .globl       prefix_sum
  .p2align     4, 0x90
  .type prefix_sum,@function
prefix_sum:
  .cfi_startproc
# %bb.0:
  pushl %ebp
  .cfi_def_cfa_offset 8
  .cfi_offset %ebp, -8
  movl  %esp, %ebp
  ...
```

llc

```
$ llc file.ll -march=arm -o file.arm
$ cat file.arm
```

```
prefix_sum:
  .fnstart
@ %bb.0:
  sub   sp, sp, #32
  str   r0, [sp, #24]
  str   r1, [sp, #16]
  str   r2, [sp, #12]
  ldr   r0, [sp, #12]
  cmp   r0, #1
  blt   .LBB0_10
  b  .LBB0_1
  ...
```

Contact & References

DCC

Fernando Magno Quintão Pereira

fernando@dcc.ufmg.br

http://lac.dcc.ufmg.br

References

https://llvm.org/docs/GettingStarted.html#an-example-using-the-llvm-tool-chain