COLUMBIA | ENGINEERING
The Fu Foundation School of Engineering and Applied Science

Introduction of Clang/LLVM

Dongdong She, Yangruibo Ding

Department of Computer Science, Columbia University

Adapted from Cornell Prof. Adrian Sampyso’s blog.

Overview

- General Introduction of Clang/LLVM.
    - What is LLVM?
    - LLVM Architecture
    - LLVM IR
    - LLVM Pass
- Program Analysis with LLVM: Example.
- Write a Function Pass
    - Translate source code into LLVM IR using Clang
    - Identify basic blocks of LLVM IR
    - Control Flow Graphs
    - PA-3: Generate and Analyze CFG with LLVM Pass

Overview

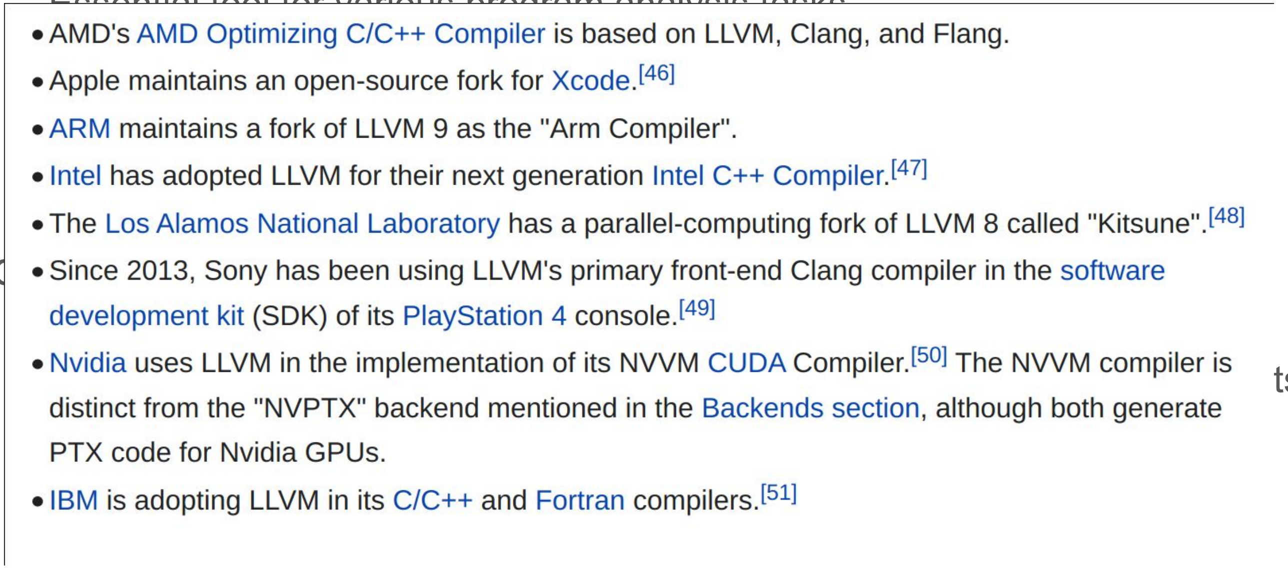
- General Introduction of Clang/LLVM.
    - What is LLVM?
    - LLVM Architecture
    - LLVM IR
    - LLVM Pass
- Program Analysis with LLVM: Example.
    - Write a Function Pass
    - Translate source code into LLVM IR using Clang
    - Identify basic blocks of LLVM IR
    - Control Flow Graphs
    - PA-3: Generate and Analyze CFG with LLVM Pass

What is LLVM

- An awesome compiler for native languages like C/C++/Swift.
    - Huge impact in both academia and industry.
    - A large amount of research works in SE and Sec are based on LLVM.
    - LLVM is widely used in industry to build real world applications.
- Static single assignment (SSA) based intermediate representation (IR).
    - **Each variable is only assigned once.**
    - Avoid any value assignment ambiguity for easier and more accurate optimization.
- Modular design, easy to hack.

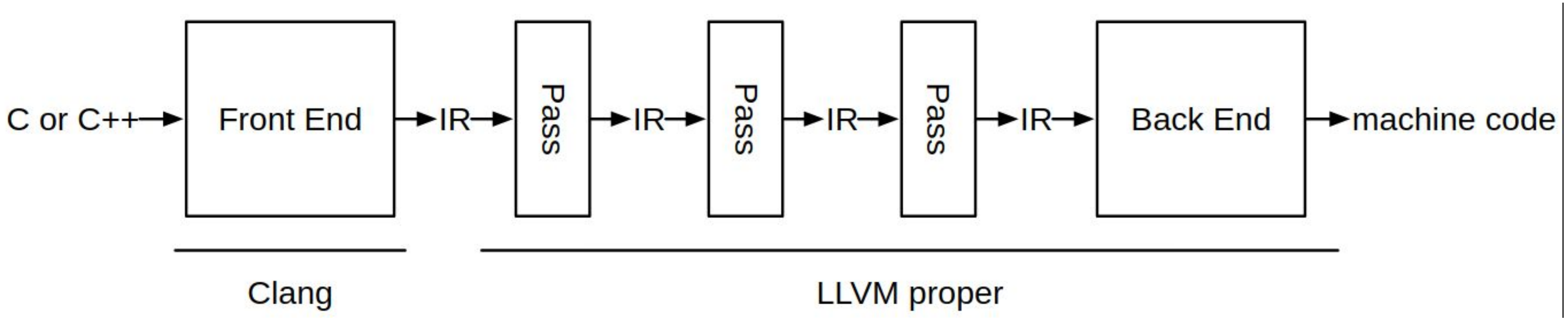
Usage of LLVM

- Academic research
  - Essential tool for various program analysis tasks
  -
  -
  -
  -
- Inc
  -
  - ts.



- AMD's AMD Optimizing C/C++ Compiler is based on LLVM, Clang, and Flang.
- Apple maintains an open-source fork for Xcode.[46]
- ARM maintains a fork of LLVM 9 as the "Arm Compiler".
- Intel has adopted LLVM for their next generation Intel C++ Compiler.[47]
- The Los Alamos National Laboratory has a parallel-computing fork of LLVM 8 called "Kitsune".[48]
- Since 2013, Sony has been using LLVM's primary front-end Clang compiler in the software development kit (SDK) of its PlayStation 4 console.[49]
- Nvidia uses LLVM in the implementation of its NVVM CUDA Compiler.[50] The NVVM compiler is distinct from the "NVPTX" backend mentioned in the Backends section, although both generate PTX code for Nvidia GPUs.
- IBM is adopting LLVM in its C/C++ and Fortran compilers.[51]

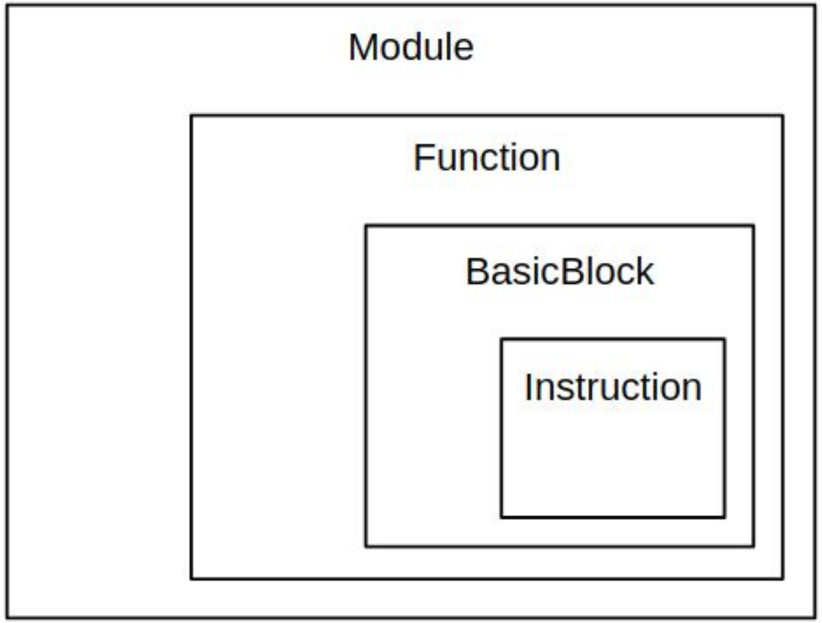
LLVM Architecture



- Front end: translate code into IR. (Your prog-hw2)
- Pass: **translate IR to IR with various optimization**. (Your future prog-hw)
- Back end: translate IR to machine code. (Rarely used in practice)
- All three components is hackable.

LLVM IR

LLVM IR Components



- Module: A single translation unit, normally equivalent to a source file.
- Function: represent a function defined in source code.
- BasicBlock: a chunk of sequentially executed instructions without branch.
- Instruction: a single code operation.

LLVM IR

LLVM IR Example

i32 %5 = add i32 %4, 2

- Opcode:
  - add: represents addition operation
- Source Operand:
  - i32 %4: a 32-bit long register named with “4”
  - 2: a number literal
- Destination Opcode:
  - i32 %5: A 32-bit long register named with “5”
- Semantic: Add %4 and 2, put results into %5.

LLVM Pass

LLVM pass transform IR to IR with various optimizations.

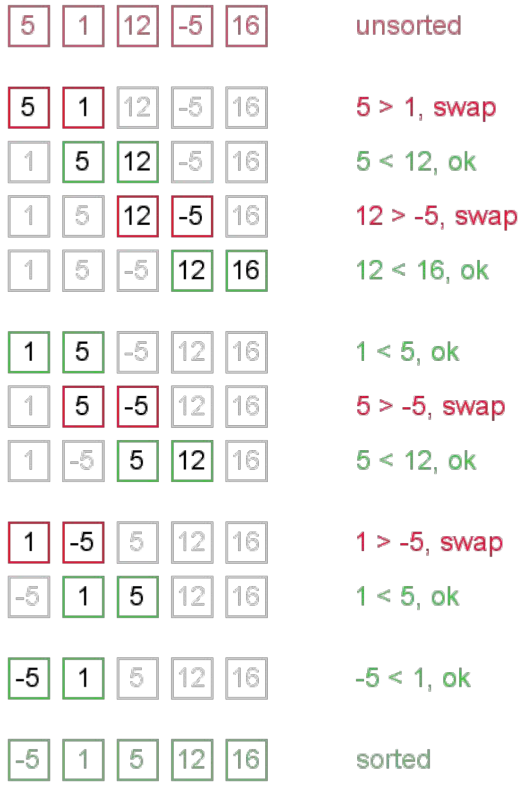
An simple LLVM Pass example: print every IR instructions.

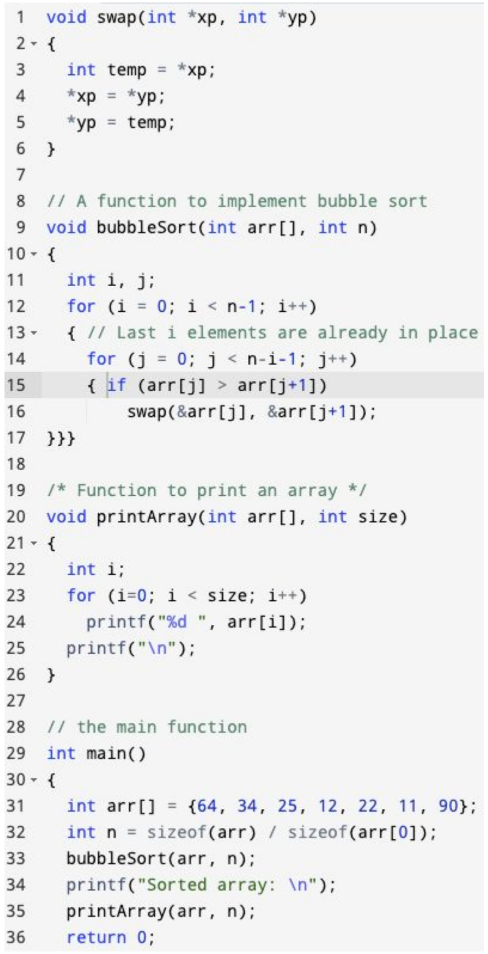
```
for (auto& F : M) {                          // iterate every function inside a module
   for (auto& B : F) {                       // iterate every BB inside a function
      for (auto& I : B) {                    // iterate every instruction inside a function
          errs() << "Instruction: " << I << "\n";     // print instruction
      }
   }
}
```

Overview

- General Introduction of Clang/LLVM.
  - What is LLVM?
  - LLVM Architecture
  - LLVM IR
  - LLVM Pass
- Program Analysis with LLVM: Example.
  - Write a Function Pass
  - Translate source code into LLVM IR using Clang
  - Identify basic blocks of LLVM IR
  - Control Flow Graphs
  - PA-3: Generate and Analyze CFG with LLVM Pass

Example: Bubble Sort





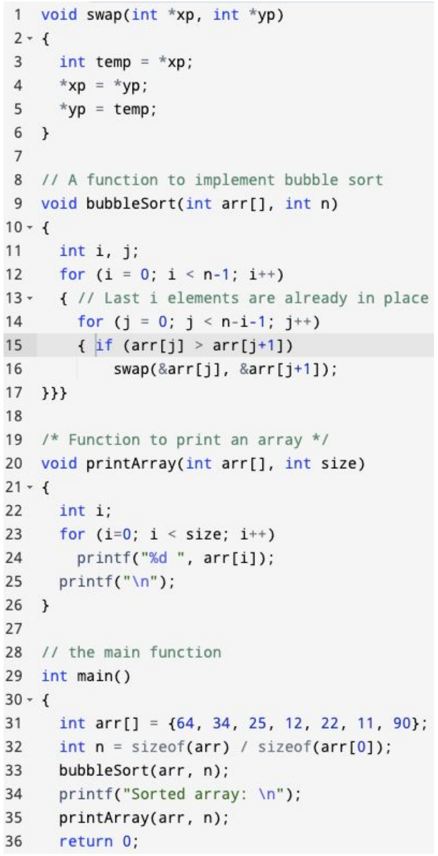
```c
void swap(int *xp, int *yp)
{
  int temp = *xp;
  *xp = *yp;
  *yp = temp;
}

// A function to implement bubble sort
void bubbleSort(int arr[], int n)
{
  int i, j;
  for (i = 0; i < n-1; i++)
  { // Last i elements are already in place
    for (j = 0; j < n-i-1; j++)
    { if (arr[j] > arr[j+1])
          swap(&arr[j], &arr[j+1]);
}}}

/* Function to print an array */
void printArray(int arr[], int size)
{
  int i;
  for (i=0; i < size; i++)
    printf("%d ", arr[i]);
  printf("\n");
}

// the main function
int main()
{
  int arr[] = {64, 34, 25, 12, 22, 11, 90};
  int n = sizeof(arr) / sizeof(arr[0]);
  bubbleSort(arr, n);
  printf("Sorted array: \n");
  printArray(arr, n);
  return 0;
```

COLUMBIA | ENGINEERING
The Fu Foundation School of Engineering and Applied Science

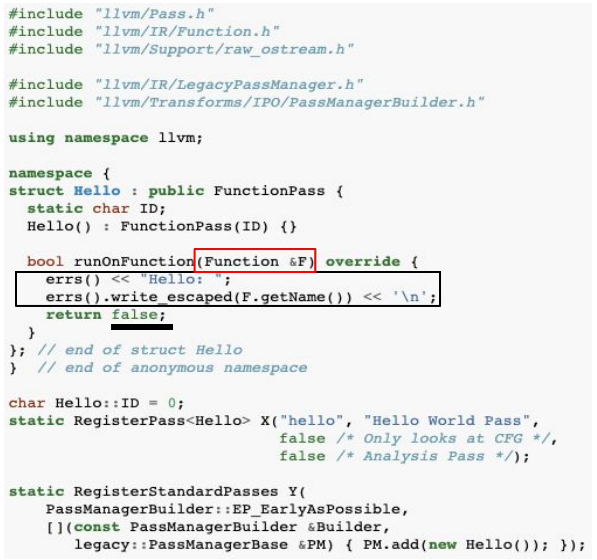
Write a Function Pass

```
#include "llvm/Pass.h"
#include "llvm/IR/Function.h"
#include "llvm/Support/raw_ostream.h"

#include "llvm/IR/LegacyPassManager.h"
#include "llvm/Transforms/IPO/PassManagerBuilder.h"

using namespace llvm;

namespace {
struct Hello : public FunctionPass {
  static char ID;
  Hello() : FunctionPass(ID) {}

  bool runOnFunction(Function &F) override {
    errs() << "Hello: ";
    errs().write_escaped(F.getName()) << '\n';
    return false;
  }
}; // end of struct Hello
}  // end of anonymous namespace

char Hello::ID = 0;
static RegisterPass<Hello> X("hello", "Hello World Pass",
                             false /* Only looks at CFG */,
                             false /* Analysis Pass */);

static RegisterStandardPasses Y(
    PassManagerBuilder::EP_EarlyAsPossible,
    [](const PassManagerBuilder &Builder,
       legacy::PassManagerBase &PM) { PM.add(new Hello()); });
```

Write a Function Pass - Hello



```c
void swap(int *xp, int *yp)
{
  int temp = *xp;
  *xp = *yp;
  *yp = temp;
}

// A function to implement bubble sort
void bubbleSort(int arr[], int n)
{
  int i, j;
  for (i = 0; i < n-1; i++)
  { // Last i elements are already in place
    for (j = 0; j < n-i-1; j++)
    { if (arr[j] > arr[j+1])
        swap(&arr[j], &arr[j+1]);
}}}

/* Function to print an array */
void printArray(int arr[], int size)
{
  int i;
  for (i=0; i < size; i++)
    printf("%d ", arr[i]);
  printf("\n");
}

// the main function
int main()
{
  int arr[] = {64, 34, 25, 12, 22, 11, 90};
  int n = sizeof(arr) / sizeof(arr[0]);
  bubbleSort(arr, n);
  printf("Sorted array: \n");
  printArray(arr, n);
  return 0;
```

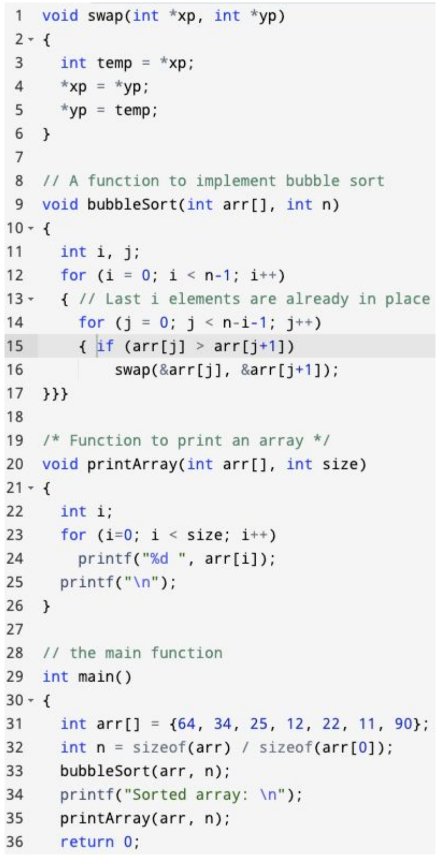


```cpp
#include "llvm/Pass.h"
#include "llvm/IR/Function.h"
#include "llvm/Support/raw_ostream.h"

#include "llvm/IR/LegacyPassManager.h"
#include "llvm/Transforms/IPO/PassManagerBuilder.h"

using namespace llvm;

namespace {
struct Hello : public FunctionPass {
  static char ID;
  Hello() : FunctionPass(ID) {}

  bool runOnFunction(Function &F) override {
    errs() << "Hello: ";
    errs().write_escaped(F.getName()) << '\n';
    return false;
  }
}; // end of struct Hello
}  // end of anonymous namespace

char Hello::ID = 0;
static RegisterPass<Hello> X("hello", "Hello World Pass",
                             false /* Only looks at CFG */,
                             false /* Analysis Pass */);

static RegisterStandardPasses Y(
    PassManagerBuilder::EP_EarlyAsPossible,
    [](const PassManagerBuilder &Builder,
       legacy::PassManagerBase &PM) { PM.add(new Hello()); });
```

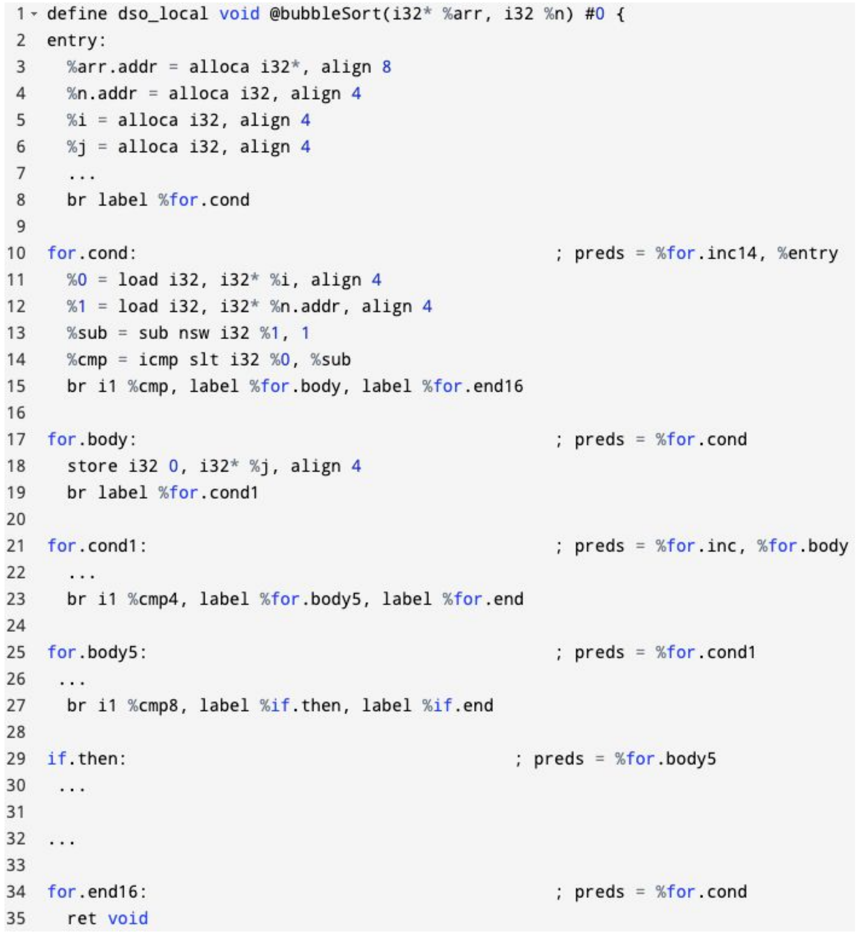
```
Hello: swap
Hello: bubbleSort
Hello: printArray
Hello: main
```

```
opt -load lib/LLVMHello.so -hello < bubble.bc
```

LLVM IR



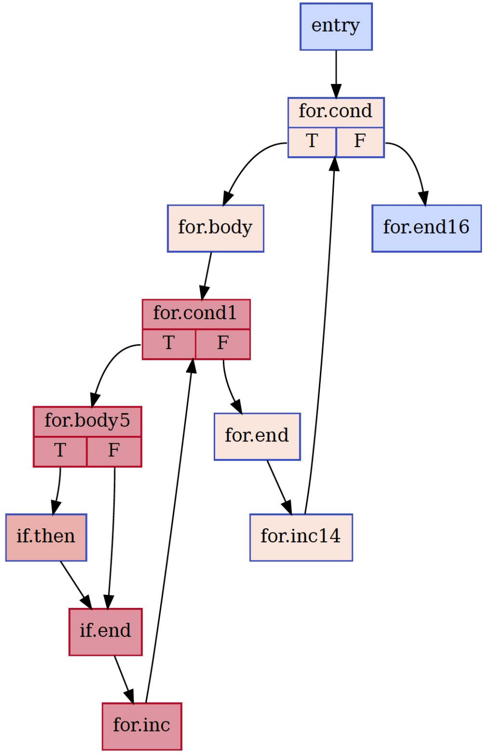
```c
void swap(int *xp, int *yp)
{
  int temp = *xp;
  *xp = *yp;
  *yp = temp;
}

// A function to implement bubble sort
void bubbleSort(int arr[], int n)
{
  int i, j;
  for (i = 0; i < n-1; i++)
  { // Last i elements are already in place
    for (j = 0; j < n-i-1; j++)
    { if (arr[j] > arr[j+1])
        swap(&arr[j], &arr[j+1]);
}}}

/* Function to print an array */
void printArray(int arr[], int size)
{
  int i;
  for (i=0; i < size; i++)
    printf("%d ", arr[i]);
  printf("\n");
}

// the main function
int main()
{
  int arr[] = {64, 34, 25, 12, 22, 11, 90};
  int n = sizeof(arr) / sizeof(arr[0]);
  bubbleSort(arr, n);
  printf("Sorted array: \n");
  printArray(arr, n);
  return 0;
```



```llvm
define dso_local void @bubbleSort(i32* %arr, i32 %n) #0 {
entry:
  %arr.addr = alloca i32*, align 8
  %n.addr = alloca i32, align 4
  %i = alloca i32, align 4
  %j = alloca i32, align 4
  ...
  br label %for.cond

for.cond:                                         ; preds = %for.inc14, %entry
  %0 = load i32, i32* %i, align 4
  %1 = load i32, i32* %n.addr, align 4
  %sub = sub nsw i32 %1, 1
  %cmp = icmp slt i32 %0, %sub
  br i1 %cmp, label %for.body, label %for.end16

for.body:                                         ; preds = %for.cond
  store i32 0, i32* %j, align 4
  br label %for.cond1

for.cond1:                                        ; preds = %for.inc, %for.body
  ...
  br i1 %cmp4, label %for.body5, label %for.end

for.body5:                                        ; preds = %for.cond1
 ...
  br i1 %cmp8, label %if.then, label %if.end

if.then:                                      ; preds = %for.body5
 ...

...

for.end16:                                        ; preds = %for.cond
  ret void
```

COLUMBIA | ENGINEERING
The Fu Foundation School of Engineering and Applied Science

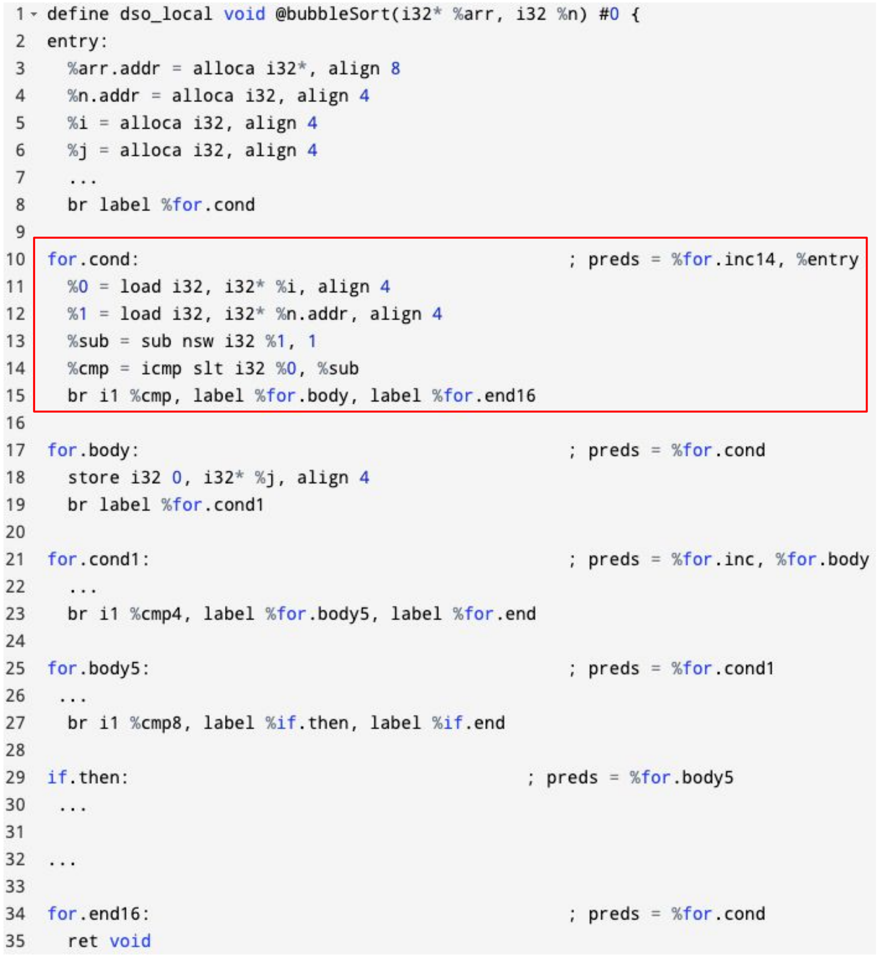
Control Flow Graph



CFG for 'bubbleSort' function

```
define dso_local void @bubbleSort(i32* %arr, i32 %n) #0 {
entry:
  %arr.addr = alloca i32*, align 8
  %n.addr = alloca i32, align 4
  %i = alloca i32, align 4
  %j = alloca i32, align 4
  ...
  br label %for.cond

for.cond:                                         ; preds = %for.inc14, %entry
  %0 = load i32, i32* %i, align 4
  %1 = load i32, i32* %n.addr, align 4
  %sub = sub nsw i32 %1, 1
  %cmp = icmp slt i32 %0, %sub
  br i1 %cmp, label %for.body, label %for.end16

for.body:                                         ; preds = %for.cond
  store i32 0, i32* %j, align 4
  br label %for.cond1

for.cond1:                                        ; preds = %for.inc, %for.body
  ...
  br i1 %cmp4, label %for.body5, label %for.end

for.body5:                                        ; preds = %for.cond1
 ...
  br i1 %cmp8, label %if.then, label %if.end

if.then:                                      ; preds = %for.body5
 ...


...

for.end16:                                        ; preds = %for.cond
  ret void
```

Basic Blocks



```
define dso_local void @bubbleSort(i32* %arr, i32 %n) #0 {
entry:
  %arr.addr = alloca i32*, align 8
  %n.addr = alloca i32, align 4
  %i = alloca i32, align 4
  %j = alloca i32, align 4
  ...
  br label %for.cond

for.cond:                                         ; preds = %for.inc14, %entry
  %0 = load i32, i32* %i, align 4
  %1 = load i32, i32* %n.addr, align 4
  %sub = sub nsw i32 %1, 1
  %cmp = icmp slt i32 %0, %sub
  br i1 %cmp, label %for.body, label %for.end16

for.body:                                         ; preds = %for.cond
  store i32 0, i32* %j, align 4
  br label %for.cond1

for.cond1:                                        ; preds = %for.inc, %for.body
  ...
  br i1 %cmp4, label %for.body5, label %for.end

for.body5:                                        ; preds = %for.cond1
 ...
  br i1 %cmp8, label %if.then, label %if.end

if.then:                                      ; preds = %for.body5
 ...


...

for.end16:                                        ; preds = %for.cond
  ret void
```

Basic Blocks

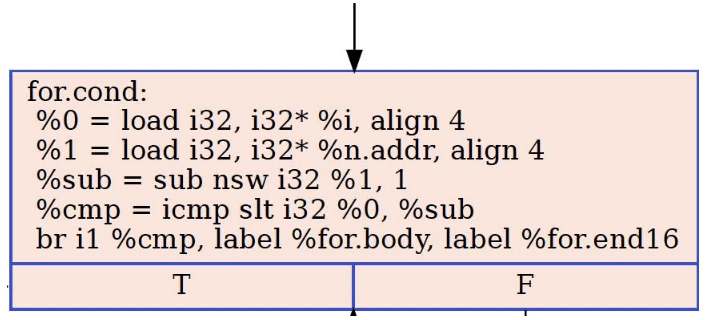
- A basic block is a straight-line code sequence with no branches in except to the entry and no branches out except at the exit.

- Compilers usually decompose programs into their basic blocks as a first step in the analysis process. Basic blocks form the vertices or nodes in a control-flow graph

Basic Blocks

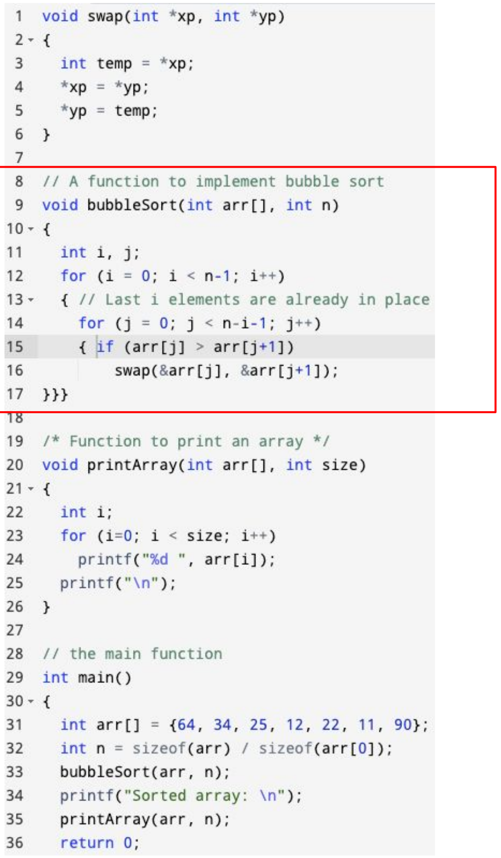
- A basic block is a straight-line code sequence with no branches in except to the entry and no branches out except at the exit.

- Compilers usually decompose programs into their basic blocks as a first step in the analysis process. Basic blocks form the vertices or nodes in a control-flow graph

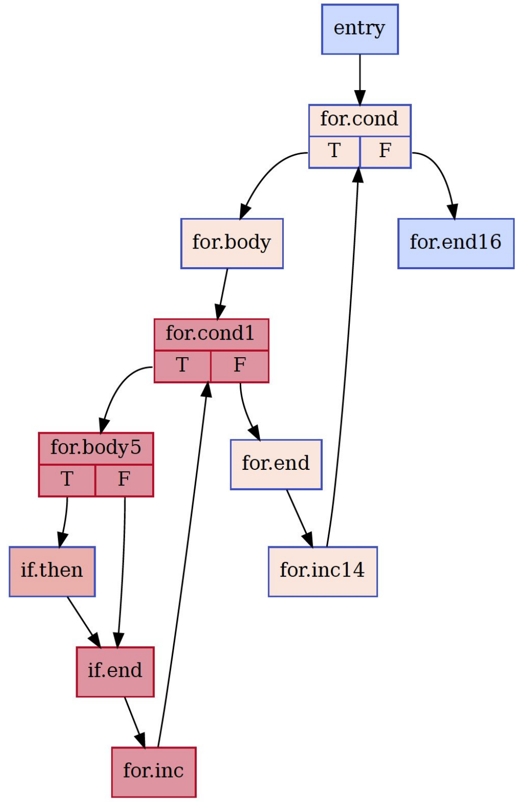


```
for.cond:
 %0 = load i32, i32* %i, align 4
 %1 = load i32, i32* %n.addr, align 4
 %sub = sub nsw i32 %1, 1
 %cmp = icmp slt i32 %0, %sub
 br i1 %cmp, label %for.body, label %for.end16
```

PA-3: Generate and Analyze CFG with LLVM Pass



```
void swap(int *xp, int *yp)
{
  int temp = *xp;
  *xp = *yp;
  *yp = temp;
}

// A function to implement bubble sort
void bubbleSort(int arr[], int n)
{
  int i, j;
  for (i = 0; i < n-1; i++)
  { // Last i elements are already in place
    for (j = 0; j < n-i-1; j++)
    { if (arr[j] > arr[j+1])
        swap(&arr[j], &arr[j+1]);
}}}

/* Function to print an array */
void printArray(int arr[], int size)
{
  int i;
  for (i=0; i < size; i++)
    printf("%d ", arr[i]);
  printf("\n");
}

// the main function
int main()
{
  int arr[] = {64, 34, 25, 12, 22, 11, 90};
  int n = sizeof(arr) / sizeof(arr[0]);
  bubbleSort(arr, n);
  printf("Sorted array: \n");
  printArray(arr, n);
  return 0;
```



CFG for 'bubbleSort' function

COLUMBIA | ENGINEERING
The Fu Foundation School of Engineering and Applied Science

PA-3: Generate and Analyze CFG with LLVM Pass

- Announcement Date: Today, Nov. 10
- Due Date: Wednesday, Nov. 24
- **LLVM version should be <= 12.x.x**
- START EARLY !!!!!