

Compiler Project Readme

Atharv Rajendra Jagtap (2101088)

Gaurav Kumar Patel (2101075)

Kushagra Patel (2101109)

Date: 04/03/2024



Problem Statement:

Design an LLVM pass to partition C++ code into regions based on power consumption, where regions are functions of the program. Also, print the ID of the desired function.

1 LLVM Overview

LLVM (Low Level Virtual Machine) is a compiler infrastructure project that provides a set of modular and reusable compiler and tool-chain technologies. It is widely used in both industry and academia.

1.1 Components of LLVM

- Front-end for different programming languages
- Mid-level Intermediate Representation (IR)
- Optimization passes
- Code generators
- Debuggers
- Just-In-Time (JIT) compiler

1.2 Features of LLVM

- Modular and reusable components
- Support for various programming languages: C, C++, Objective-C, Swift, Rust, etc.
- Flexibility to target a wide range of hardware platforms
- Extensibility for custom tool-chains and optimizations

2 Project Details

2.1 Assumptions

- a) Energy consumption is measured using the Energy-interference-free Debugger (EDB) with voltage watchpoints.
- b) We already have assumed (approximated with respect to time required for each instruction) power values for operations like add, sub, etc.

2.2 Actual Methodology

- **Energy Measurement Setup:** Utilize EDB connected to the capacitor on the target device to record capacitor voltage at watchpoints.
- **Voltage Watchpoints Placement:** Place voltage watchpoints in the code to mark start (V_{from}) and end (V_{to}) of sections to measure energy consumption.
- **Energy Calculation:** Calculate energy consumed between watchpoints using the formula $E = \frac{1}{2}C(V_{from}^2 - V_{to}^2)$.
- **Effective Capacity Calculation:** Compute effective capacity using V_{on} and V_{off} voltages for accurate energy calculations.
- **Measurement Precision:** Ensure precise voltage measurements using EDB for accurate energy consumption calculations.
- **Power Calculation:** Calculate average power by dividing energy by execution time for accurate power representation.

2.3 Reference paper

You can access the research paper related to this project by following this link:

<https://dl.acm.org/doi/pdf/10.1145/3178372.3179525>

3 Steps to Install and Run LLVM

3.1 Install LLVM

If you are using a Linux-based operating system, you can install LLVM and Clang using the following commands:

```
sudo apt-get install llvm clang
```

3.2 Verify Installation

After installation, verify that LLVM and Clang are correctly installed by checking their versions:

```
llvm-config --version  
clang --version
```

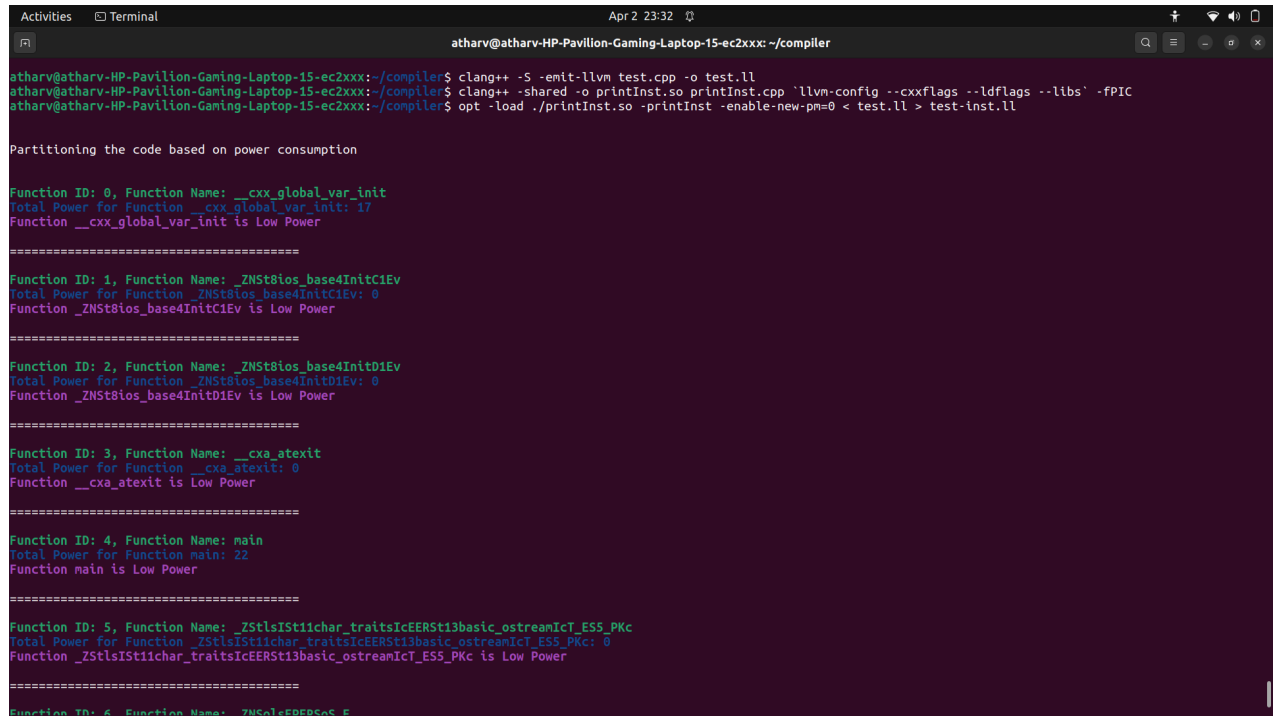
3.3 Running LLVM Commands

Once LLVM and Clang are installed, you can use the following commands to compile and run LLVM programs:

```
clang++ -S -emit-llvm test.cpp -o test.ll  
clang++ -shared -o printInst.so printInst.cpp 'llvm-config --cxxflags --ldflags --libs' -flto  
opt -load ./printInst.so -printInst -enable-new-pm=0 < test.ll > test-inst.ll
```

These commands demonstrate compiling C++ code to LLVM IR, creating a shared object file, and running an LLVM pass on the LLVM IR file.

4 Output



```
athary@athary-HP-Pavilion-Gaming-Laptop-15-ec2xxx: ~/compiler
clang++ -S -emit-llvm test.cpp -o test.ll
clang++ -shared -o printInst.so printInst.cpp `llvm-config --cxxflags --ldflags --libs` -fPIC
opt -load ./printInst.so -printInst -enable-new-pm=0 < test.ll > test-inst.ll

Partitioning the code based on power consumption

Function ID: 0, Function Name: __cxx_global_var_init
Total Power for Function __cxx_global_var_init: 17
Function __cxx_global_var_init is Low Power
=====
Function ID: 1, Function Name: _ZNSt8ios_base4InitC1Ev
Total Power for Function _ZNSt8ios_base4InitC1Ev: 0
Function _ZNSt8ios_base4InitC1Ev is Low Power
=====
Function ID: 2, Function Name: _ZNSt8ios_base4InitD1Ev
Total Power for Function _ZNSt8ios_base4InitD1Ev: 0
Function _ZNSt8ios_base4InitD1Ev is Low Power
=====
Function ID: 3, Function Name: __cxa_atexit
Total Power for Function __cxa_atexit: 0
Function __cxa_atexit is Low Power
=====
Function ID: 4, Function Name: main
Total Power for Function main: 22
Function main is Low Power
=====
Function ID: 5, Function Name: _ZStlsISt11char_traitsIcEERSt13basic_ostreamIcT_ES5_Pkc
Total Power for Function _ZStlsISt11char_traitsIcEERSt13basic_ostreamIcT_ES5_Pkc: 0
Function _ZStlsISt11char_traitsIcEERSt13basic_ostreamIcT_ES5_Pkc is Low Power
=====
Function ID: 6, Function Name: _ZN6pl5F0E85a5 E
```

Figure 1: First Image

```
Activities Terminal Apr 2 23:32
atharv@atharv-HP-Pavilion-Gaming-Laptop-15-ec2xxx: ~/compiler

=====
Function ID: 5, Function Name: _ZStlsISt11char_traitsIcEERSt13basic_ostreamIcT_E55_PKc
Total Power for Function _ZStlsISt11char_traitsIcEERSt13basic_ostreamIcT_E55_PKc: 0
Function _ZStlsISt11char_traitsIcEERSt13basic_ostreamIcT_E55_PKc is Low Power
=====

Function ID: 6, Function Name: _ZNSolsEPFRSoS_E
Total Power for Function _ZNSolsEPFRSoS_E: 0
Function _ZNSolsEPFRSoS_E is Low Power
=====

Function ID: 7, Function Name: _ZSt4endlIcSt11char_traitsIcEERSt13basic_ostreamIT_T0_ES6_
Total Power for Function _ZSt4endlIcSt11char_traitsIcEERSt13basic_ostreamIT_T0_ES6_: 0
Function _ZSt4endlIcSt11char_traitsIcEERSt13basic_ostreamIT_T0_ES6_ is Low Power
=====

Function ID: 8, Function Name: _Z5func2v
Total Power for Function _Z5func2v: 40
Function _Z5func2v is High Power
=====

Function ID: 9, Function Name: _ZNSolsEi
Total Power for Function _ZNSolsEi: 0
Function _ZNSolsEi is Low Power
=====

Function ID: 10, Function Name: _Z9functioniv
Total Power for Function _Z9functioniv: 735
Function _Z9functioniv is High Power
=====

Function ID: 11, Function Name: _GLOBAL__sub_I_test.cpp
Total Power for Function _GLOBAL__sub_I_test.cpp: 9
Function _GLOBAL__sub_I_test.cpp is Low Power
=====

atharv@atharv-HP-Pavilion-Gaming-Laptop-15-ec2xxx:~/compiler$
```

Figure 2: Second Image

5 Conclusion

By implementing the described methodology and using LLVM infrastructure, we can determine the power consumption of specific operations . This approach is beneficial for fine-grained energy profiling and optimization in embedded systems.