

Indian Institute of Information Technology, Allahabad

**PROJECT REPORT**

horizontal line

LOW POWER OPTIMIZATION FOR

EMBEDDED COMPILERS

**Project Supervisor - Dr. Bibhas Ghoshal**

**Declaration by the Candidates**

We, hereby declare that the project titled *Low Power Optimization for Embedded Compilers* is a record of bonafide project work carried out by us under the guidance of *Dr. Bibhas Ghoshal* in partial fulfillment of the 5th semester Mini-Project work for the B.Tech (IT) Course in Indian Institute of Information Technology, Allahabad.

Nishit Gupta – IIT2014502

Sachin Agarwal – IIT2014501

D. Rajeswar Rao – IIT2014055

Saurabh Tanwar – IIT2014140

**Certificate**

This is to certify that the project report entitled *Low Power Optimization for Embedded Compilers* submitted to Department of Information Technology, Indian Institute of Information Technology, Allahabad in partial fulfillment of the 5th semester Mini-roject work, is a record of bonafide work carried out by:

1. Nishit Gupta – IIT2014502
2. Sachin Agarwal – IIT2014501
3. D. Rajeswar Rao – IIT2014055
4. Saurabh Tanwar – IIT2014140

under my supervision and guidance.

This report has not been submitted anywhere else for any other purpose.

Submission Date : 01/12/2016

**Dr.Bibhas Ghoshal**

Assistant Professor

Department of Information Technology

Indian Institute of Information Technology

Allahabad - 211012

**CONTENTS**

|  |  |  |
| --- | --- | --- |
| S.No. | Topic | Page No |
|  | Introduction | 5 |
|  | Problem Statement and Objective | 6 |
|  | Literature Survey | 7 |
|  | Proposed Approach | 8-10 |
|  | Software & Hardware Requirements | 11 |
|  | Activity Chart | 12 |
|  | Work Completed Pre Mid-Semester | 13-17 |
|  | Work Completed Post Mid-Semester | 18-21 |
|  | Results | 22 |
|  | Conclusion | 23 |
|  | Future Scope | 23 |
|  | References | 24 |

# Introduction

1. Background

An optimizing compiler tries to **minimize** or **maximize** some attributes of an executable computer program.

The most common requirement is to **minimize** the

* **time** taken to execute a program,
* the **amount of memory** occupied and
* the **power** consumed by a program.

1. Motivation

The present focus of Compiler Optimization techniques is reducing the “Execution Time” of the program.

With the advancement in technology, portable devices, embedded systems etc. are being widely developed which are handy and easy to use. These portable devices are battery powered.

Along with performance, the power consumption also needs to be looked at i.e. we need devices which should consume less power along with generating output faster.

Therefore, the idea of **Low Power Optimization.**

Energy has become an important design consideration, together with performance in computer systems.

While hardware optimizations has been the focus of several studies and are fairly mature, software approaches to optimizing power are relatively new.

Progress in understanding the impact of traditional compiler optimizations and developing new power-aware compiler optimizations are important to overall system energy optimization.

Here we focus on two questions?

* + Is the most efficient code from Execution Time perspective same as that for the Energy viewpoint?
  + If not, then what are the Power-Aware Optimization Techniques?

We conducted a few experiments to test the foretasted theory on ARM Architecture using Gem5 Simulator.



From above experiment we arrived at the conclusion that power optimization being a by-product of compiler optimization did not hold true and hence we now pave the way for the Problem Statement and Objective.

# Problem Statement and Objective

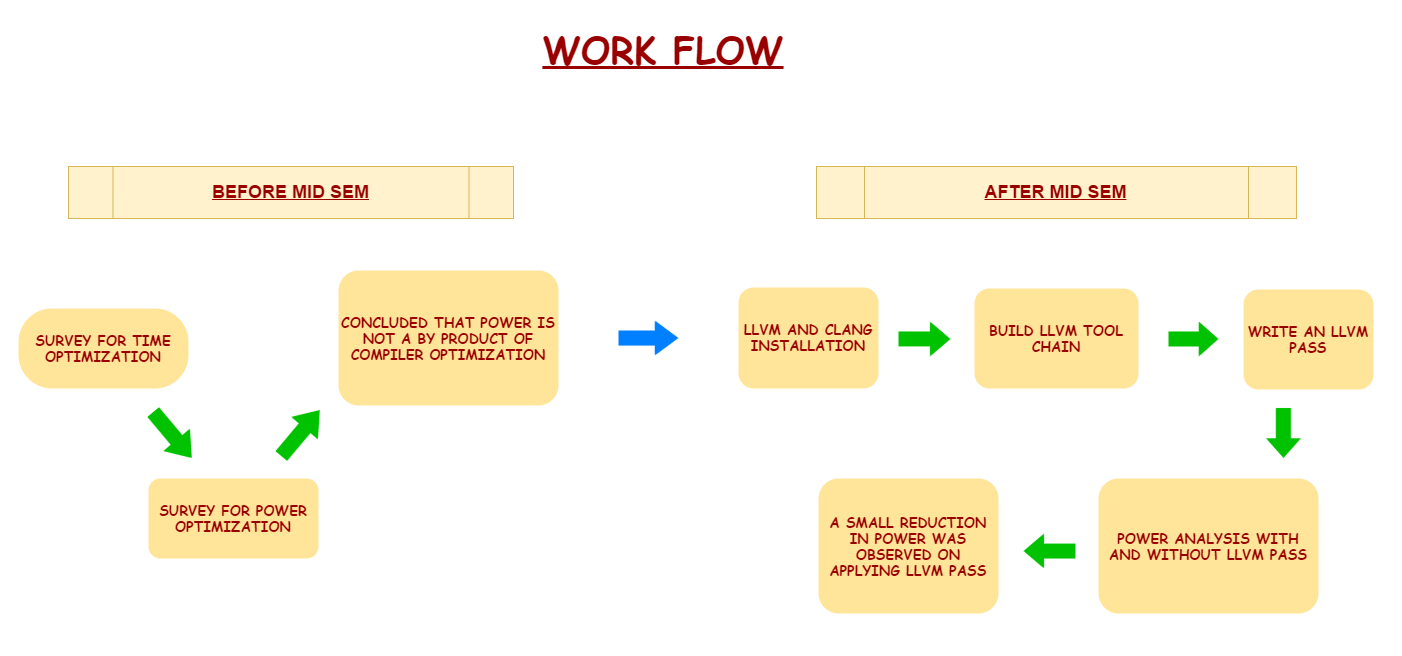
1. Exploring various compiler optimization techniques (GCC) and study their effect on power optimization.
2. Develop low power compiler techniques as patches for gnu tool chain.

**Literature Survey**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **S No.** | **Author** | **Paper Title** | **Year** | **Crux** | **Venue** |
|  | David Branco & P.R Henriques | *Impact of GCC Optimization levels in energy consumptions during C/C++ program execution. [1]* | 2015 | *Presenting experimental setup and method followed to measure and compare resources consumed by a program during execution.* | *2015 IEEE 13th International Scientific Conference on Informatics.* |
|  | Vivek Tiwari, Sharad Malik and A. Wolfe | *Compilation techniques for low energy : an overview[2]* | 1994 | *Used techniques such as Re-ordering instructions to reduce switching and using patterns for Code generations to reduce Power.*  *Conclusion: Conducted an experiment which reduced power upto 40%.* | In *Low Power Electronics, 1994, Digest of Technical Papers, IEEE Symposium.* |
|  | M Kandemir, N Vijaykrishnan and M.J. Irwin | *Power aware computing[3]* | 2002 | *Focuses on two power aware low-level techniques: 1) Instruction Scheduling for reducing switching acitivity, and 2) Post-compilation relabeling of Register for reducing Power.* | In *Chapter Compiler Optimizations for Low Power Systems* |
|  | M Valluri and Lizy K. John | *Is Compiling for Performance -- Compiling for Power? [4]* | 2001 | *They present a quantitative study where they examine the effect of the standard optimizations levels −01 to −04 on power and energy of the processor. They also evaluate the effect of four individual optimizations on power/energy and classify them as “low energy” or “low power” optimizations.* | Springer, USA, Boston, MA |
|  | U. Kremer | *Low Power/Energy Compiler Optimizations[5]* | 2005 | *Comparison of Power & Energy and Performance Analysis and concluded that both of them are different strategies and one can not be a by-product of the other.* | *Low-Power Electronics Design, CRC Press, 2005* |

**Proposed Approach**

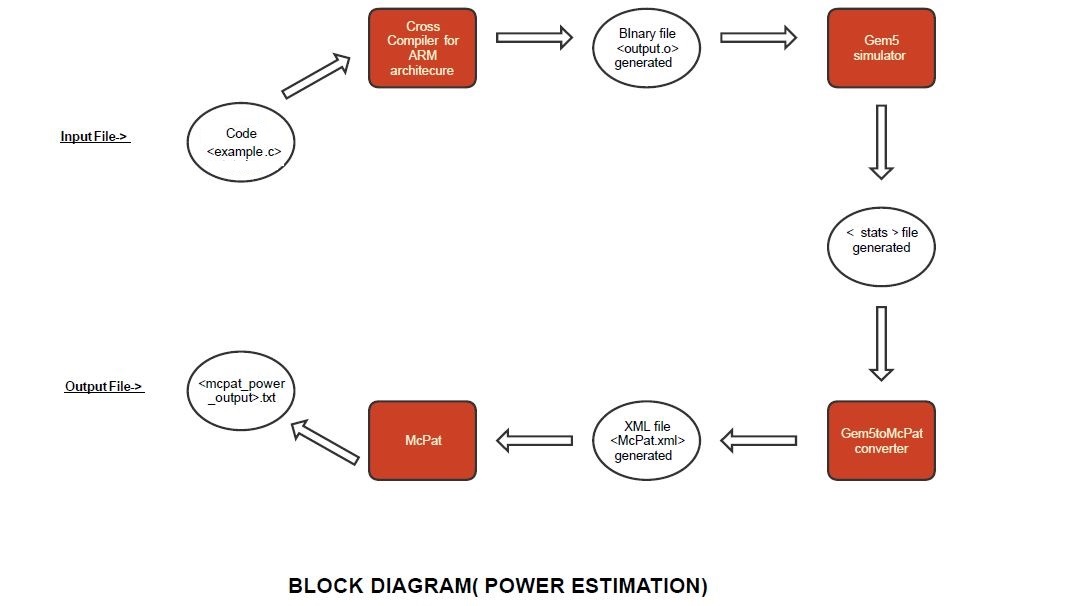
**Project Workflow Approach**



**Module 1 – Execution Time Survey**

**Module 2 – Power Survey**

**Module 2 – Power Estimation**

****

**Module 3 – Develop a Pass for Power Optimization**

****

**Software & Hardware Requirements**

**Software Requirements:**

1. Gem5
2. ARM architecture Cross Compiler
3. Gem5ToMcpatConverter*[6]*
4. McPat*[7]*
5. LLVM 3.7.1*[8]*
6. CLANG Compiler*[9]*
7. GNUPlot
8. GCC (C language for programs)
9. Ubuntu 14.04/ Linux
10. CMake*[10]*

**Hardware Requirements:**

Testing Platform : Laptop HP ENVY-J106TX, running under Linux and Windows10

**Hardware/Software Specifications:**

Architecture : 64bit system

Operating System : Ubuntu 16.04 64-bit / Windows 10

Microprocessor : 2.8 GHz Intel Core i5-6610U

Memory : 12 GB 1600 MHz DDR4L SDRAM (1 x 4 GB + 1 x 8 GB)

**Minimum Requirement**

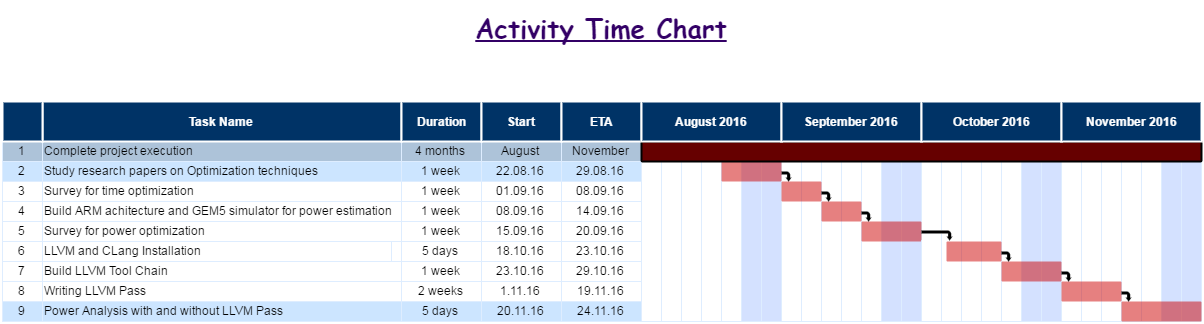
Microprocessor :- Intel i5 4th Gen 2.4Ghz or above

Memory: 2GB or above

Hard Disk : 100 GB

Operating System: Windows / Linux

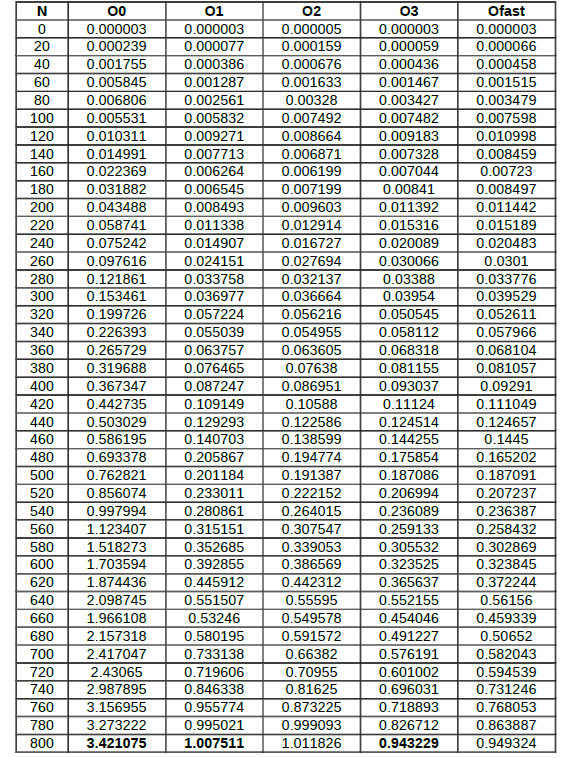
**Activity Time Chart**



**Work completed Pre Mid-Semester**

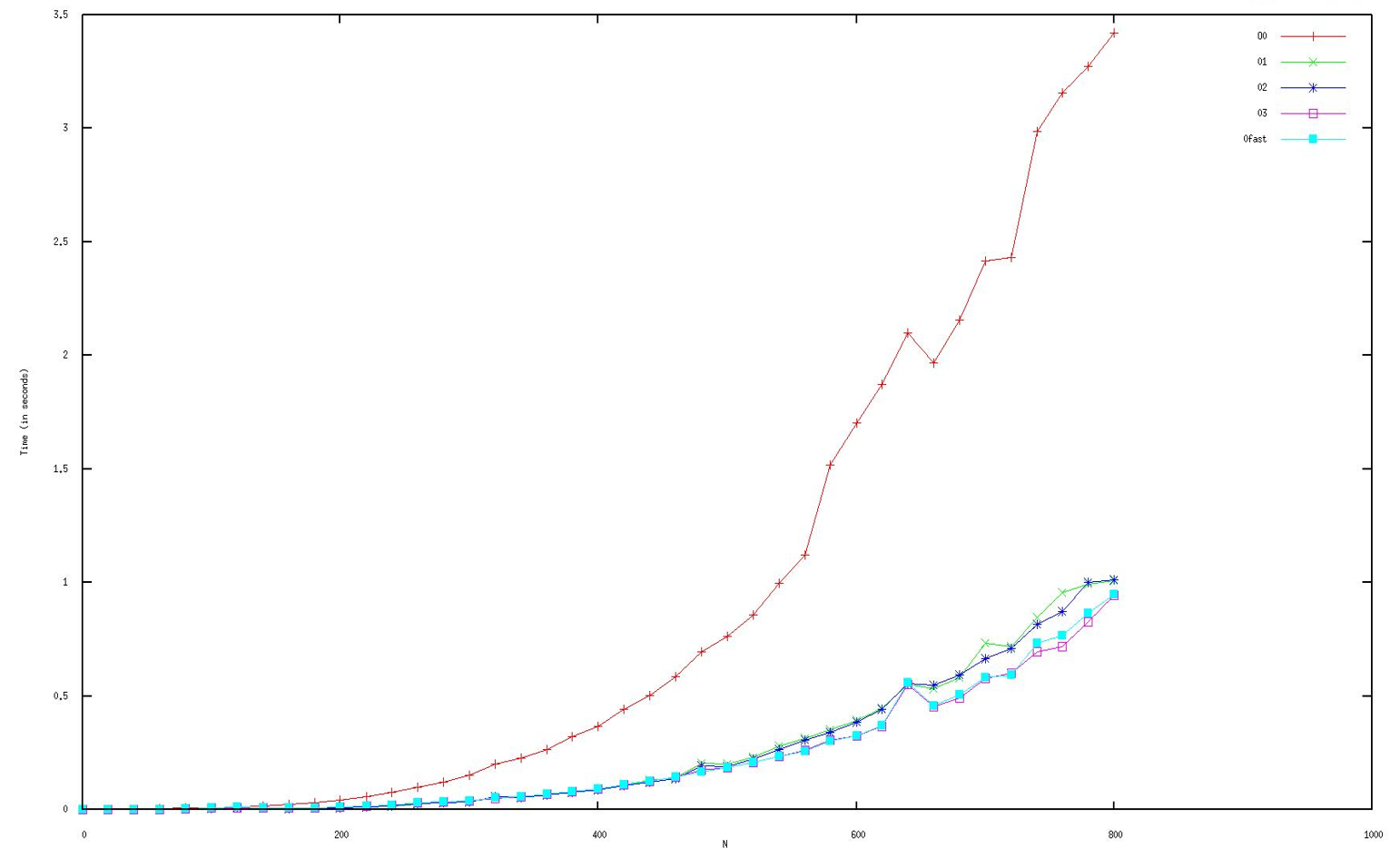
**Matrix Multiplication**

**Comparison of Execution Time**



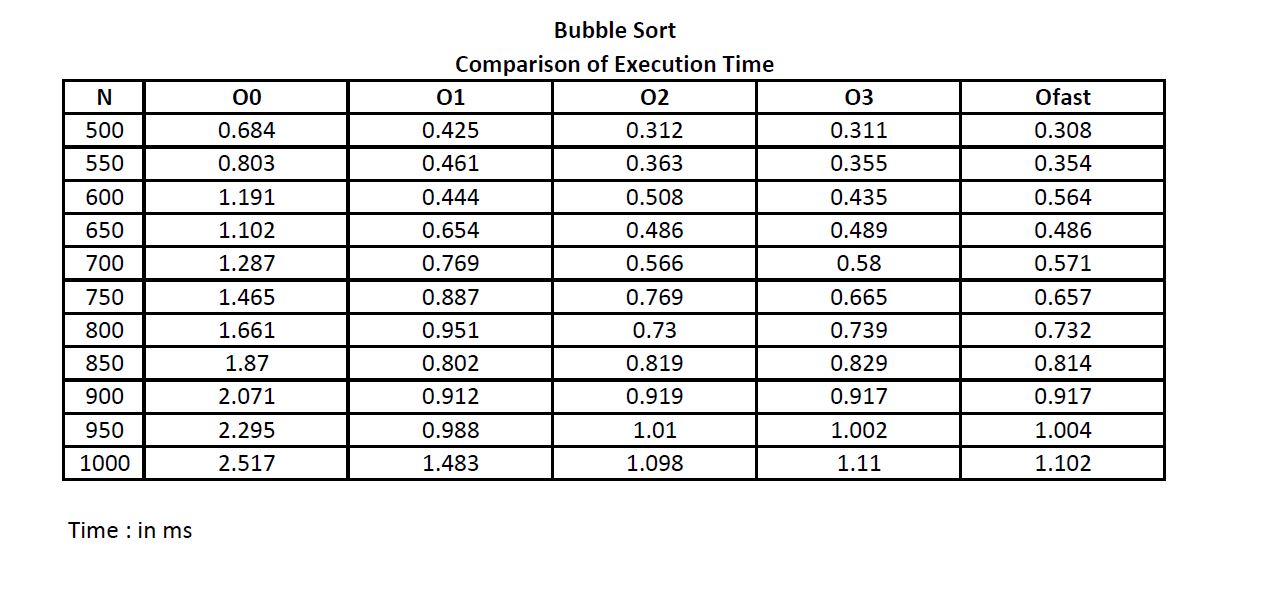
**Matrix Multiplication**

**Graph of Execution Time**



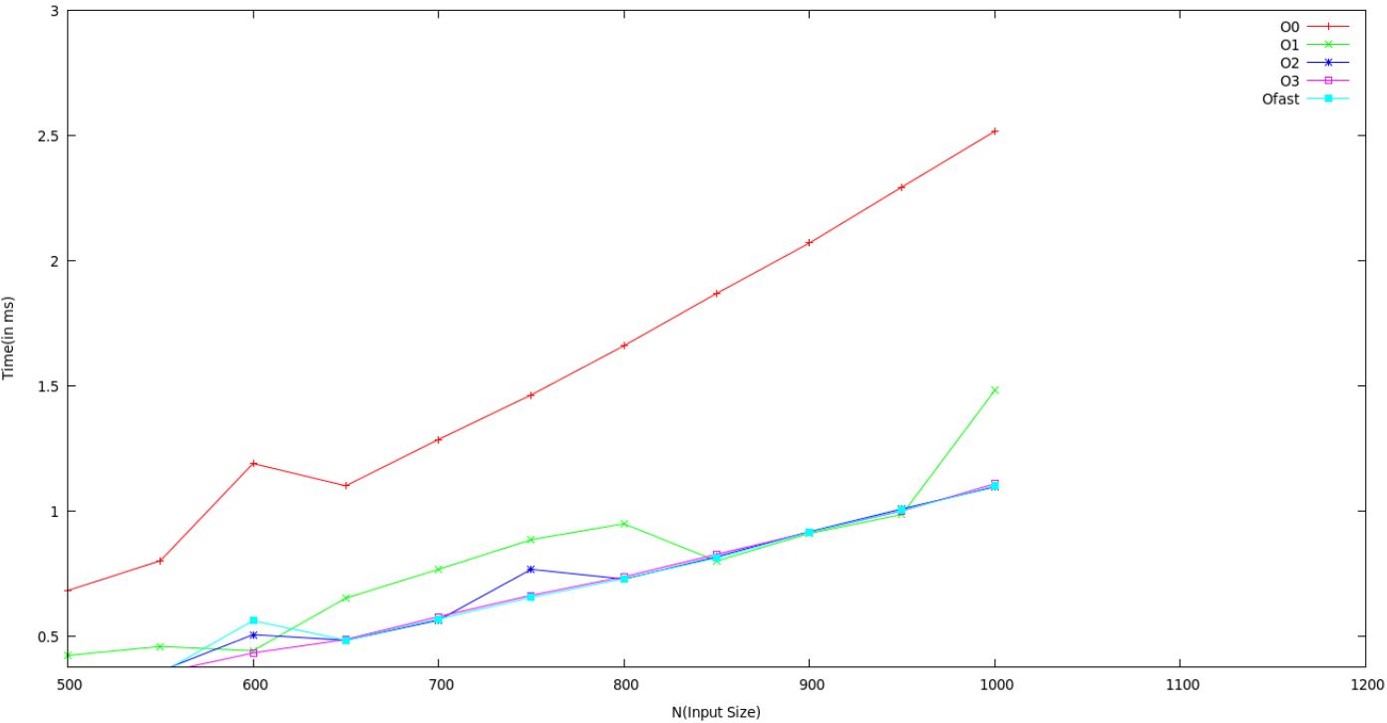
**Bubble Sort**

**Comparison of Execution Time**

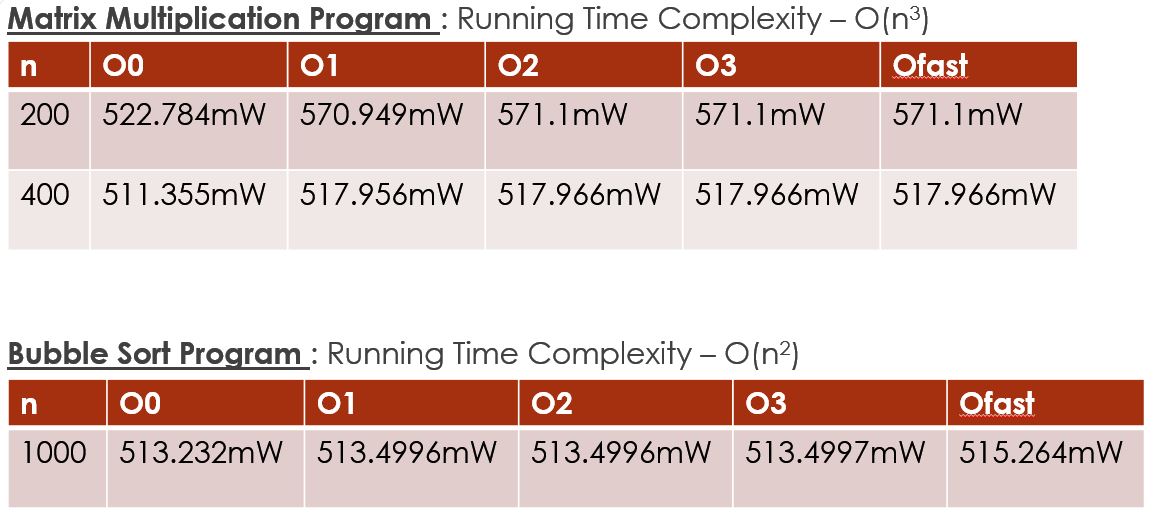


**Bubble Sort**

**Graph of Execution Time**



**Comparison of Power Consumption**



**Work completed Post Mid-Semester**

**1.** Building of LLVM Tool Chain

**2.** Study of Various Power Aware Optimization Techniques

**3.** Writing of LLVM Pass

**4.** Application of Pass on Matrix Multiplication Program

**LLVM pass**

* Passes are the small pieces of code that implement the optimizations and transformations for the compiler
* Passes structure the compiler code and build analysis results used by these transformations
* The Pass class is the superclass of the LLVM Passes
* These passes override virtual methods which they inherit from Pass
* LLVM Pass Framework is that it schedules passes to run in an efficient way based on the constraints that your pass meets.

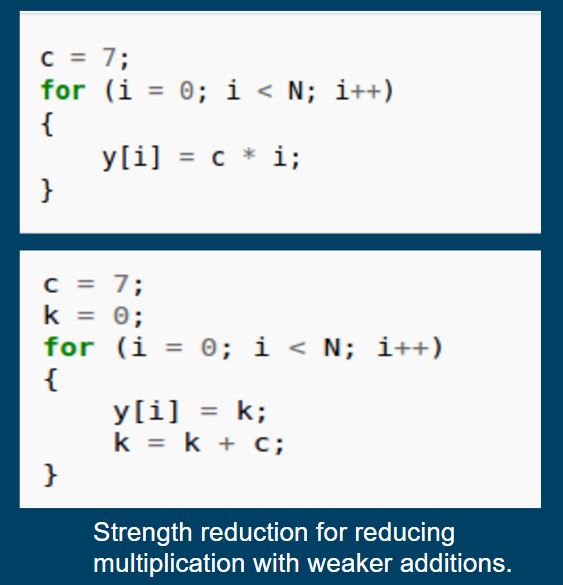
Some of the Power-Aware Optimizations include:

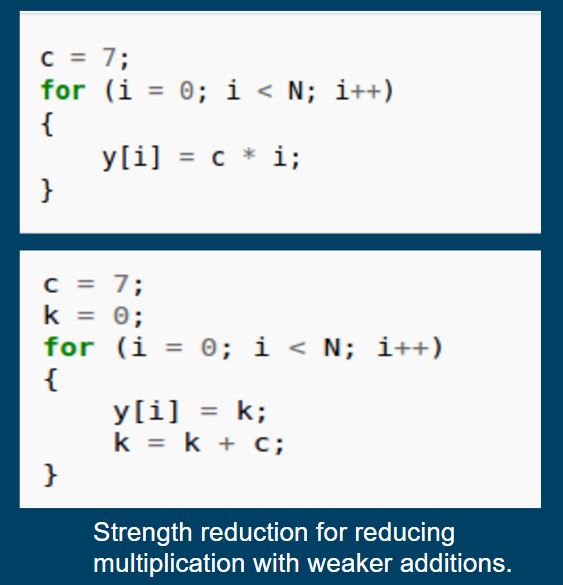
* **Tuning of Memory Optimizations** 
  + Memory sub-sytems - considered one of the most energy consuming parts of devices
  + Minimization of memory accesses is an important approach in lowering power consumption
  + A few existing memory optimizations include prefetching, automatic vectorization, linear loop transformation etc.
  + **PREFETCHING**
    - Used to speed up the execution of a program by reducing wait states.
    - Occurs when a processor requests an instruction or data block from main memory before it is actually needed.
    - reduces power by only searching the prefetch hardware tables for selective memory instructions identified by the compiler
  + **AUTOMATIC VECTORIZATION**
    - Automatic vectorization is an optimization that converts several scalar operations into one vector operation.
    - As ARM supports loading of several registers from memory simultaneously in one operation, this optimization potentially offers decrease in memory access operations.
* **Reordering instructions to reduce switching**
  + Switching activity in a circuit is a function of the present inputs and the previous state of the circuit.
  + It is expected that energy consumed during execution of a particular instruction will vary depending on what the previous instruction was.
  + Thus an appropriate reordering of instructions in a program can result in lower energy.
  + A few known approaches include : gray code technique, cold scheduling
  + **GRAY CODE TECHNIQUE**
    - Consecutive gray codes have only one bit difference in consecutive numbers
    - Gray code addressing can significantly reduce the number of bit switches.
    - Experimental results show that using Gray code addressing reduce the switching activity at the address lines by 30~50% compared to using normal binary code addressing.
  + **COLD SCHEDULING TECHNIQUE**
    - Described in (C. L. Su et al. Low power architecture design and compilation techniques for high-performance processors. In IEEE COMPCON, Feb. 1994.)
    - Is a software method which schedules instructions in a way that switching activity is minimized.
    - Uses traditional performance-driven scheduling techniques with special heuristics.
    - Results from experiments with cold scheduling on the VLSI-BAM show that switching activity in the control path is reduced by 20-30%.

**Strength Reduction Optimization**

Strength Reduction is a compiler optimization where expensive operations are replaced with less expensive operations but these operations are equivalent to previous expensive operations.

The basic example of Strength Reduction is that weaker additions can replace strong multiplications.

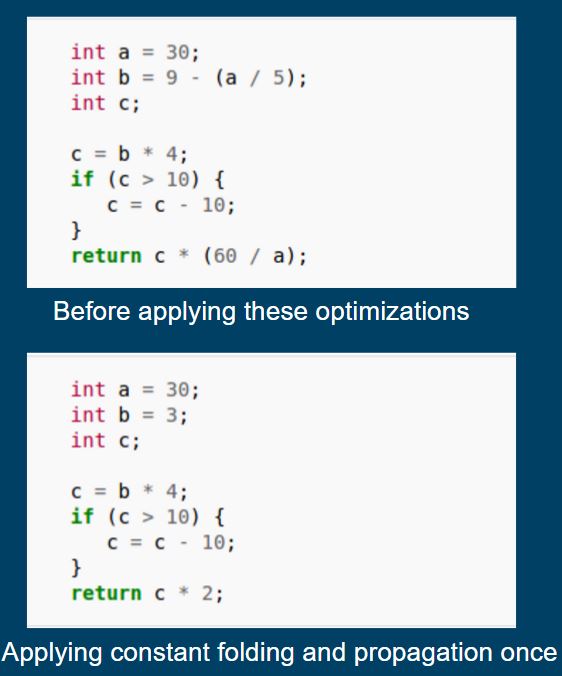
****

****

**Constant Folding Optimization**

It is the process of identifying and computing the constant expressions at compile time rather than computing them at the run-time.

Constant Propagation is the process of putting the known values of the constants in the expression during the compile time only rather than waiting to do it in the run-time.

****

**Arithmetic Optimizations**

* + x + 0 = 0 + x = x
  + x - 0 = x
  + 0 - x = -x
  + x/x = 1
  + x \* 1 = 1 \* x = x
  + x/1= x
  + x \* 0 = 0 \* x = 0

i.e. We will check both the operands and the operator and if they satisfy the above mentioned conditions, then we will directly return the result without actually computing + , - , \* or / thereby reducing the instructions to be executed and also a CPU-task.C

If Statement is

A \* = ( B / B )

Here we will save two arithmetic operations per statement.

So if this statement is executed a large number of times (as in the case of Matrix Multiplication) then there will be a significant reduction in the number of Instructions.

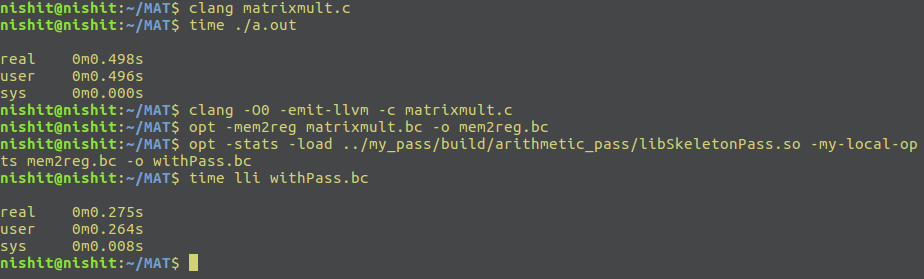
**Results**

The developed pass reduces both:

1. Execution Time
2. Power Consumed

Here are the statistics for Execution Time and Power

**Execution Time**

**Matrix Multiplication**

**Comparison of Power**

% Reduction in execution time = 44.7 %

**Power Consumed**

|  |  |
| --- | --- |
| **WITHOUT PASS** | **WITH PASS** |
| PROCESSOR | % Reduction = 22.13 % |
| Runtime Dynamic : 0.658171 W | Runtime Dynamic : 0.512492 W |
| CORE | % Reduction = 22.23 % |
| Runtime Dynamic : 0.658171 W | Runtime Dynamic : 0.511807 W |

**Conclusion**

FIRST PHASE

On the basis of execution time survey and power survey conducted the conclusion was that the present optimization techniques merely focus on execution time reduction and not on power reduction

Hence there is need to explore optimization techniques whose primary goal is to reduce power consumption.

SECOND PHASE

After going through various power optimization techniques the most favorable power optimization techniques include:

A) Instruction Rescheduling

B) Tuning Memory Optimizations

C) Reducing switching

D) Constant Folding and Constant Propagation

E) Strength Reduction

To implement any optimization writing an LLVM pass is required. LLVM generates an IR code on which the pass incorporates the specified optimizations on the code and produces the result.

We implemented a pass which targeted constant folding and strength reduction optimizations focusing on algebraic operations thereby reducing instruction count and succeeded in reducing power as opposed to the increase in power consumption by previously existent optimization techniques.

**Future Scope**

* + - 1. This project can further be extended making use of Heuristic Function for more reduced Power

Consumption.

* + - 1. More Optimization Techniques can be implemented in a single Pass for further reducing power and time

both.

1. Reduced Power Consumption can pave a bright future path for Embedded Systems and other Portable

Battery-Operated Devices.

**References**

[1] D Branco and P.R Henriques: Impact of GCC Optimizations Levels in Energy Consumption during C or C++ program execution.

2015 IEEE 13th International Scientific Conference on Informatics.

[2] V.Tiwari, S Malik and A. Wolfe : Compilation Techniques for Low Energy, An Overview.

-Low Power Electronics, 1994. Digest of Technical Papers, IEEE Symposium.

Pages 38–39, Oct 1994.

[3] M. Kandemir, N Vijaykrishnan and M. Jane Irwin: Power Aware Computing, Chapter: Compiler Optimizations for Low Power Systems.

-Pages : 191 to 210. Kluwer Academic Publishers, Norwell, MA, USA – 2002

[4] M. Valluri and Lizy K. John: Is Compiling for Performance – Compiling for Power?

Springer, USA, Boston, MA – 2001

[5] U Kremer: Low Power/Energy Compiler Optimizations

In Low-Power Electronics Design, CRC Press, 2005

[6] https://bitbucket.org/dskhudia/gem5tomcpat

[7] http://www.hpl.hp.com/research/mcpat/

[8] http://llvm.org/

[9] http://clang.llvm.org/

[10] https://cmake.org/