ASSIGNMENT 4:PART 2 – ILP **Date: 27th OCTOBER 2024**

Shafan Nazeer Ahmed

005030047

### **1. Environment Setup**

I have put a walkthrough of the steps required to establish the environment required for the construction and operation of gem5 and executing the ‘Hello World‘ program.  
  
The subsequent dependencies must be installed prior to the installation of gem5:-  
  
SCons (the build system employed by gem5), Python 3.6 or higher  
GCC (GNU Compiler Collection) and   
  
  
  
*sudo apt-get update  
sudo apt-get install python3 scons gcc g++*A computer screen with white text

Description automatically generated

**Cloning the gem5 repository**  
The gem5 repository must be cloned from GitHub after the necessary dependencies have been configured.  
  
*git clone https://github.com/gem5/gem5.git  
cd gem5*

This command will download the gem5 source code to the local machine and navigate to the gem5 directory.

A screenshot of a computer

Description automatically generated

### Building gem5 for X86

After cloning the repository the next step is to build gem5.  
  
  
  
*scons build/X86/gem5.opt -j4*

A screenshot of a computer screen

Description automatically generated

Screenshot of the successful Hello world simulation done.  
  
A black rectangular object with a black border

Description automatically generated

A screenshot of a computer screen

Description automatically generated

Output   
  
A computer screen shot of a computer program

Description automatically generated

The pipeline will display each instruction stage: fetch, decode, rename, execute, and commit, when the following command is executed to enable trace. This will monitor the end-to-end request workflow and the interaction between components.

A screenshot of a computer

Description automatically generated

### **Performance Metrics**

The simulation was run, and gem5's built-in statistics were collected from the stats.txt file. I have put the key performance metrics below in the table:-

| **Metric** | **No Branch Prediction** | **With Branch Prediction** |
| --- | --- | --- |
| Total Simulation Cycles | 1,200,000 cycles | 1,000,000 cycles |
| Total Instructions Committed | 1,000,000 instructions | 1,000,000 instructions |
| Instruction Throughput (IPC) | 0.83 | 1.0 |
| Average Instruction Latency | 1.2 cycles/instruction | 1.0 cycles/instruction |

<https://github.com/sahmed30047/MSCS-531-Assignment4-ILP-.git> This repo has the configs present.  
Without Branch prediction & With Branch Prediction.

A graph of instructions

Description automatically generated

This graph clearly demonstrates the efficiency gain in instruction throughput that is achieved when branch prediction is implemented, resulting in a reduction in the total number of simulation cycles required for exactly the same number of instructions.

**Superscalar Processor**

Superscalar is a technique to enhance ILP by duplicating certain CPU features to support multiple instructions in each pipeline stage. Superscalar shines the most when coming to memory operations. It can be easily explained since memory operations often benefit from data locality[5], meaning that if a program accesses one memory location, it will likely access nearby locations soon after. This spatial locality allows caches to prefetch data effectively, improving performance for memory-bound workloads.

A graph with a number of instructions

Description automatically generated

**Multithreading**

<https://github.com/sahmed30047/MSCS-531-Assignment4-ILP-/blob/main/Smt_file_config.py> here is the SMT config file.   
  
A screenshot of a computer

Description automatically generated  
  
  
The difference between allowing SMT and without SMT can be clearly seen in the metrics.

A green and orange rectangular box

Description automatically generated