Computer Architecture Assignment Report

- (Non-pipelined and Pipelined MIPS processor design)

A. Lokesh - IMT2022577

A. Nishith – IMT2022556

Non-pipelined:

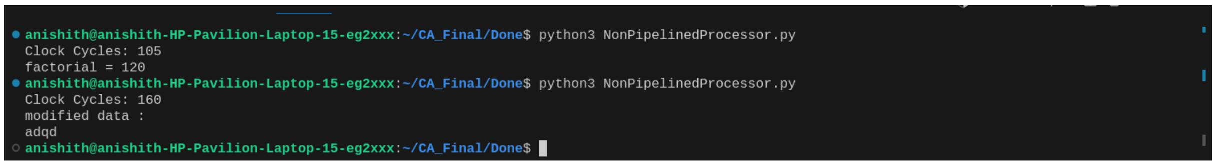
Explanation of the code:

We have maintained an array for memory, register file, and machine code. This program consists of a function "ALU" for ALU operations using a set of if-else conditions which does corresponding operation wrt the opcode. "Control_unit" function in the code sets the control signals by mapping the opcodes. There are functions like "write_back", "reg_read", "mem_write", "mem_read" and "ins_fetch" which does their usual operations when called.

The program decodes the machine code read from the file and returns the values to respective functions. Program maintains the counter ("clock") for number of clock cycles for the non-pipelined processor, 5 gets added to the "clock" for every instruction.

The programs that we chose to implement are:

1) Encryption and decryption
2) Factorial



```
● anishith@anishith-HP-Pavilion-Laptop-15-eg2xxx:~/CA_Final/Done$ python3 NonPipelinedProcessor.py
  Clock Cycles: 105
  factorial = 120
● anishith@anishith-HP-Pavilion-Laptop-15-eg2xxx:~/CA_Final/Done$ python3 NonPipelinedProcessor.py
  Clock Cycles: 160
  modified data :
  adqd
○ anishith@anishith-HP-Pavilion-Laptop-15-eg2xxx:~/CA_Final/Done$ 
```

Pipelined:

Explanation of the code:

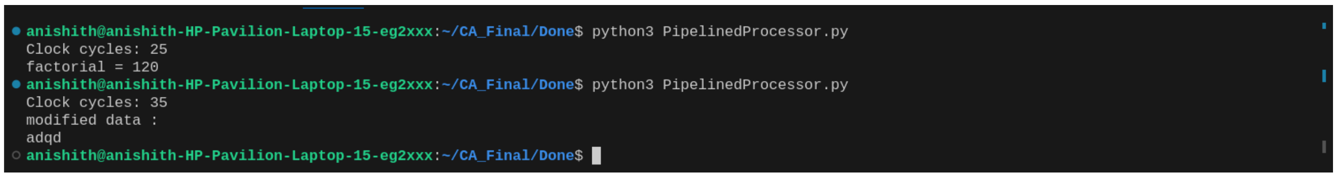
In this program we have declared array of machine code, reg_file, memory, and dictionary of IF_ID, ID_EX, EX_MEM, MEM_WB. “ALU” function in the code with set of if-else conditions will perform respective operations if mapped by opcodes. There is a “control_unit” which sets the control signals using instruction format.

There are functions like “write_back”, "reg_read”, “mem_write”, “mem_read” and “ins_fetch” which does their usual operations when called. The program decodes the machine code read from the file and returns the values to respective functions

The program detects dependencies by using conditional statements. Our machine code contains data and control hazards. We have solved the hazards using forwarding and control hazards using flushes. And the clock has been increased by 1 every time for counting the number of cycles.

The programs that we chose to implement are:

1) Encryption and decryption
2) Factorial



```
● anishith@anishith-HP-Pavilion-Laptop-15-eg2xxx:~/CA_Final/Done$ python3 PipelinedProcessor.py
  Clock cycles: 25
  factorial = 120
● anishith@anishith-HP-Pavilion-Laptop-15-eg2xxx:~/CA_Final/Done$ python3 PipelinedProcessor.py
  Clock cycles: 35
  modified data :
  adqd
○ anishith@anishith-HP-Pavilion-Laptop-15-eg2xxx:~/CA_Final/Done$
```