Term Project Submission 2 - Student Names

- C Akshay Kumar (192SP006)
- K Kiran Kumar (192SP012)

Progress and Ideas for Future Work

- As stated in the Future work section of our report 1 we have successfully achieved these things:
  - The problem of implementation is resolved and the design is now ready to be **synthesized**.
  - We have added **Data forwarding and BTB** to the processor.
  - We have designed a assembler which converts a given assembly level language into machine code. It was designed in verilog and it helps us in verifying our processor with big programs.
  - Not only a assembler we have also creater a Program Generator which will help us generate random programs and judge the working of the processor in much detail.
  - We have upgraded our 4 instruction ISA to a decent ISA of **27** Instructions including all kinds of instructions like R-Type, I-type and J-type.
  - All kinds of Pipeline Hazards and dependencies were handled with the corresponding solutions given for the same in the lectures.
- In our final report we are trying to add **Floating Point Arithmetic** and **Multi Cycle DLX** discussed in class to our processor and hence making the design **SUPER SCALAR**.

* Term project Submission 1 is added at the end of this report for easy reference.

Programs Used For Verification

LEVEL	**PROGRAM NAME**	**STALLING** *(cycles)*	**DATA FORWARDING** *(cycles)*	**DF and BTB** *(cycles)*
1	SWAPPING	26	11	11
2	MULTIPLICATION (5*10)	42	27	20
2	FACTORIAL (5!)	132	84	71
3	MATRIX MULTIPLICATION (3 ×3)	324	258	221
3	POWER X,Y (5 ^10)	474	312	278
4	SORTING AN ARRAY (6 elements)	644	363	338

* We wantedly removed Multiplication and Division instructions from our ISA to used them as loops.

Conclusion Drawn from the Results

We are using 6 General Purpose Programs of 4 different difficulty levels which will make use of all the supported instructions and we have tabulated the performance in terms of cycles for Stalling, Data Forwarding and BTB. When we observe the trend, we can understand that a processor without Data Forwarding unit will waste a lot of cycles due to data hazards. So it is must to use a DF unit to avoid these stalls, but BTB is more like a enhancement needed while we have more and more loops in the program. A processor with a DF unit and BTB will definitely attain the best performance.

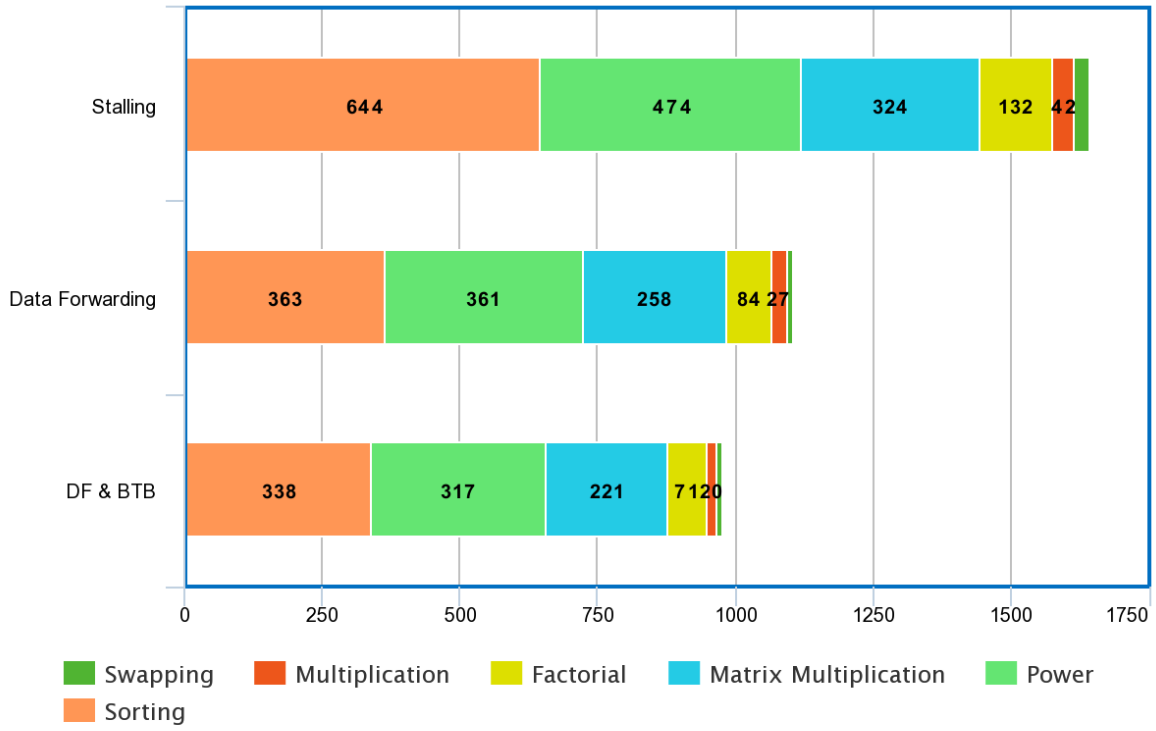

Figure 1: Performance Graph between Stalling, DF and DF with BTB

Supported Instructions

INSTRUCTION	OPCODE	FUNCTION	MEANING	CLASS	TYPE
ADD	6'b000000	6'b000001	Rx = Ry + Rz	BINARY	R
SUB	6'b000000	6'b000010	Rx = Ry - Rz	BINARY	R
AND	6'b000000	6'b000100	Rx = Ry & Rz	BINARY	R
OR	6'b000000	6'b001000	Rx = Ry |Rz	BINARY	R
XOR	6'b000000	6'b010000	Rx = Ry ^Rz	BINARY	R
SLT	6'b000000	6'b100000	1 if Rx <Ry	BINARY	R
ADDI	6'b001000	-	Rx = Ry + Imm	BINARY	I
SLTI	6'b001010	-	Rx = 1 if Ry <Imm	BINARY	I
ANDI	6'b001100	-	Rx = Ry & Imm	BINARY	I
ORI	6'b001101	-	Rx = Ry |Imm	BINARY	I
XORI	6'b001110	-	Rx = Ry ^Imm	BINARY	I
LW	6'b100011	-	Rx← Mem	BINARY	I
SW	6'b101011	-	Rx → Mem	BINARY	I
BEQ	6'b000100	-	Branch if Equal	BINARY	I
BNE	6'b000101	-	Branch if not Equal	BINARY	I
BLE	6'b000110	-	Branch if Lesser	BINARY	I
BGE	6'b000111	-	Branch if Greater	BINARY	I
J	6'b001010	-	Jump	-	J
ABS	6'b000001	6'b000001	|Rx |	UNARY	R
INVS	6'b000001	6'b000010	Invert Sign	UNARY	R
MOV	6'b000001	6'b000100	Rx ← Ry	BINARY	R
LS	6'b000001	6'b001000	Rx← $Ry \ll 1$	UNARY	R
RS	6'b000001	6'b010000	Rx← $Ry \gg 1$	UNARY	R
COMP	6'b000001	6'b100000	Rx	UNARY	R
INC	6'b000001	6'b100001	Rx++	UNARY	R
DCR	6'b000001	6'b100010	Rx—	UNARY	R
ROFR	6'b000001	6'b100011	R[Rx]	BINARY	R

Term Project Submission 1 - Student Names

- C Akshay Kumar (192SP006)
- K Kiran Kumar (192SP012)

Progress

- We have created a Multi Cycle MIPS Processor using procedural assigments in a single block.
- As of now, the ISA is just 4 Instructions ADD, SUB, AND and OR.

Specifications

- Five Stage pipelining with **IF, ID, EXE, MEM and WB**
- **32**-bit Program Counter.
- **32** Registers of each **32** bit size.
- **4** Pipeline Registers of IFID-64, IDEX-127, EXMEM-69, MEMWB-37 bits respectively to store the **data between blocks**.
- **3** Pipeline Registers of EX-11, MEM-4 and WB-1 bits respectively to store **control signals**.
- **32 bit** wide instruction and data memory of size 1024.
- Posedge Triggered clock of Time period = **20ns**.
- Main Control which generates control signals is kept in **ID stage** itself.

Problems Faced

- Since we are using procedural assignments there is a problem while implementation.
- We are giving the instructions into the instruction memory by manually converting them into bits and assigning them before the first pipeline block starts.

Future work

- We will try to modify the code so that it can be Implemented.
- Find a dynamic way of storing instructions into memory given the code.
- We will design a ISA with all kinds of instructions.
- To improve the throughput we will deploy **Data Forwarding**, **Branch Target Buffer** etc.
- If time permits, we will also try to design basic compiler that can **Unroll**, **Rename** and try to solve some dependencies in the code given.