Final Report Group 1

Helton Chen Claus Choy Andrew Jesson Meng Yin Tao

***Abstract*—The abstract goes here.**

I. Solution

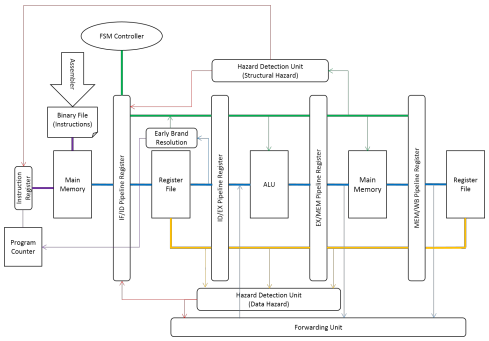

Fig. 1. Pipeline Processor Block Diagram

II. Components

The system constitutes of two components: an assembler and a processor. The assembler translates the MIPS assembly code into binary instructions. The instructions are store in the main memory along with the data. The processor accesses the memory to fetch instruction, load data and store data. The processor is pipelined into five stages and uses hazard detection, forwarding unit and early branch detection to reduce stalling time.

A. Assembler

TABLE I
Instruction Table Result of Parsing fib.asm

Mnemonic	Type	Arg 1	Arg 2	Arg 3	Block 1	...	Block N
addi	I	$10	$0	4			
addi	I	$1	$0	1			
addi	I	$2	$0	1			
addi	I	$11	$0	2000			
addi	I	$15	$0	4			
addi	I	$3	$2	0	loop		
add	R	$2	$2	$1	loop		
addi	I	$1	$3	0	loop		
mult	R	$10	$15		loop		
mflo	R	$12			loop		
add	R	$13	$11	$12	loop		
sw	I	$2	$13	0	loop		
addi	I	$10	$10	-1	loop		
bne	I	$10	$0	-9	loop		
beq	I	$11	$11	-1			

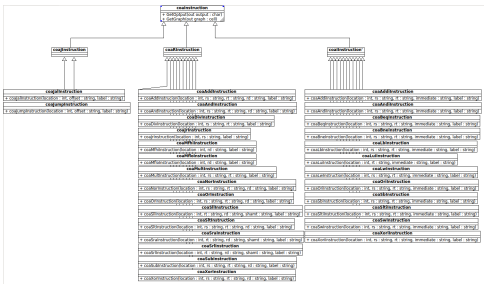

Fig. 2. Class Hierarchy of Assembler Instruction Objects

B. Instruction Fetch (IF) Stage

The first stage of the pipeline consists of fetching the instruction from memory and update the program counter (PC). MORE TO ADD...

C. Instruction Decode (ID) Stage

Once the instruction is fetched from memory, it is decoded in the ID stage, where the OPCODE is sent to the Finite-State Machine (FSM) controller. Depending on the current OPCODE and the historical state, the FSM controller will issue a series of control signals. Depending on the type of the MIPS instruction (R-type, I-type or J-type), the register file puts the corresponding data onto the register data line. To reduce stalling after a branch instruction, we implemented an early branch resolution unit.

– FSM Controller should probably belong here

D. EX Stage

E. MEM Stage

F. WB Stage

G. Pipeline

– Pipeline Registers
– Hazard Detection Unit
– Forwarding Unit
– Early Branch Resolution

H. Optimization

– If we are optimizing the processor, we need to discuss the design choices

III. System Evaluation

– How did we test it?
– What was the clock frequency?
– What was the performance?
– Dump memory and compare result against expectation?

IV. Recommendation

– Helton got this part — –intermediate steps, smaller deliverables During the first deliverable, we found it challenging when attempting to verify the results from the assembler. This is because we were not provided the expected value of the intermediate results e.g. the binary code after the assembly code is translated to bit code. The same could be said about the deliverable in general.

V. Conclusion

The conclusion goes here.

References

[1] H. Kopka and P. W. Daly, *A Guide to LaTeX*, 3rd ed. Harlow, England: Addison-Wesley, 1999.