|  |
| --- |
| uw-madison-logo-flush-web  ECE552 – Project Report  Charlie Jungwirth  Steve Akpojisheri  Peter Davis  Shane O’Donnell |

Introduction

This project entailed designing a 16 bit WISC-S25 microprocessor. Over the semester the project was split into 3 phases which documented the certain milestones which were to be met at certain times. These phases creates an overview of how the project was transformed and maintained whilst more intricate features were added. The three phases can be seen documented below:

Phase 1: This phase explored creating the baseline of the microprocessor which involved creating core modules such as the ALU in which would decide what arithmetic would take place depending on the opcode sent to the ALU. These different arithmetic functions and their opcode can be seen in figure 1.

Phase 2: This phase utilised the baseline which was created in phase 1 and implemented 5 stage pipelining into the system. This entailed using registers to store necessary values for each instruction inbetween the 5 stages: IF, ID, EX, MEM, and WB. This pipelined system would work on a predict-not-taken branch handling system as well as data hazard forwarding (EX to MEM, MEM to EX, and MEM to MEM). As well as this, a global flush/stall logic was also implemented to ensure in order completion of instructions.

Phase 3:

|  |  |
| --- | --- |
| Instruction | Opcode |
| ADD | 0000 |
| SUB | 0001 |
| XOR | 0010 |
| RED | 0011 |
| SLL | 0100 |
| SRA | 0101 |
| ROR | 0110 |
| PADDSB | 0111 |
| LW | 1000 |
| SW | 1001 |
| LLB | 1010 |
| LHB | 1011 |
| B | 1100 |
| BR | 1101 |
| PCS | 1110 |
| HLT | 1111 |

**Responsibilities/Task Breakdown**

|  |  |  |
| --- | --- | --- |
| Phase 1 | Charlie |  |
|  | Peter |  |
|  | Steve |  |
|  | Shane |  |
| Phase 2 | Charlie |  |
|  | Peter |  |
|  | Steve |  |
|  | Shane |  |
| Phase 3 | Charlie |  |
|  | Peter |  |
|  | Steve |  |
|  | Shane |  |

**Special Features**

Report any special features, any particular effort you have made to  
optimize the design, and any major problems you encountered in implementing the  
design

**Completeness**

In terms of completeness, this subject can be broken into the three phases outlining how our design compared to the desired conditions. The following phases can be seen outlined below.

**Phase 1:** Comparing to the desired functionality of phase 1, our design matched the required specifications. The ALU successfully called the correct operation according to the opcode that was sent. Each instruction needed for the ALU was instantiated inside the ALU module in which a case statement would chose the applicable output depending on the opcode sent. This allowed for the ALU output (ALU\_out) to be outputted from the module and back to the top module (cpu.v) as seen as the desired instructions instantiation has completed its instructions and sent its output to an intern wire inside the ALU module. All ALU instructions needed for this phase operated as expected, these instructions are outlined in figure 1 above.

As well as this, in this phase the memory instructions, load word, save word, load lower bits, and load higher bits. The load word instruction successfully loaded register rt with the content which was stored at the location specified in the rs register which is added the the immediate offset. The system also successfully saved words to a memory address specified in rs plus the offset.

In terms of the control instructions, the necessary instructions to implement were Branch, Branch Register, PCS, and Halt. These instructions were successfully implemented ensuring that the branch command would successfully jump to the correct address obtained by incrementing the immediate offset that was specified by 2.

**Phase 2:** The overall goal for phase 2 was to transform the foundational design created in phase 1 into a 5 stage pipelined processor. This new design would implement the 5 stages IF, ID, EX,MEM and WB. For this phase, the pipelining was successfully implemented by creating registers between each stage of the processing cycle in which current instructions and operations would be held intermediately whilst waiting for the next clock cycle. Although this design would increase latency on startup, the overall throughput would decrease, allowing the processor to perform instructions in a more efficient manner.

Phase 3:

**Testing**

A description of the methodology you used to test the correctness of your  
design.

**Results**

A summary of your simulation results for all of the test programs. Please  
include a table that shows the outcome (correct/incorrect) and cycle count for each  
program. Include a discussion of any incorrect outcomes, explaining where the  
source of the problem lies. If you made changes that improved your results after the  
demo, please note them here. Include simulation waveform to prove that