**Project Report**

**WISC-S15**

PHASE 1

ECE/CS 552

April 9, 2015

Submitted to: Parmesh Ramanathan

Graham Nygard

Robert Wagner

*INTRODUCTION*

WISC-S15 is a 16-bit computer with a load/store architecture, register file, three-bit FLAG register, reset, and fourteen arithmetic, load/store, and control instructions. The memory system is comprised of a 64Kword main memory, instruction cache, and simplified data cache. For this intermediate project report, we are to assume that the instruction cache size is as large as the main 64Kword memory with a one clock cycle access delay. Furthermore, we are also to assume that all accesses to the instruction cache will result in a hit. The second and final phase of this project will involve modifying this memory architecture to a support a direct-mapped instruction cache.

In our approach, our team focused on engineering a modular and robust Verilog architecture while achieving support for all instructions in the given ISA specification. We prioritized the development of critical test benches (such as the ALU, control logic, top level, etc.) such that we could implement a well-designed and functionally correct architecture by the deadline. Our team plans to extend our test bench framework for the remainder of the semester to increase the depth and breadth coverage of our modules.

**Special Features**

Our architecture does not currently support any additional features outside of the instructions mentioned in the Requirement Satisfaction section below. However, we plan expand to expand upon our design by implementing features such as data forwarding and auxiliary instructions not required in the original ISA specification.

**Optimizations**

Although our design does not currently express any explicit optimizations, our team wrote Verilog conscientiously in order to decrease area (e.g avoiding latches), minimize delays, and enhance the robustness and coding practices used in our architecture. When delayed branching is implemented, we will optimize our design by utilizing instruction frequencies listed in Table 3 of the project specification.

**Challenges**

The grand challenge in engineering the WISC-S15 was managing the complexity of the overall design. Our team chose to resolve this problem through the abstraction of components into a hierarchy of layers, with the top layer instantiating nine modules—five stages linked together with four pipeline registers. This approach allowed our team to break down the project into smaller units, but generated the new challenge of integrating these sub-modules as we added support for more features of the architecture. Our team responded by constructing an architecture diagram (illustrated below) in order to aid our design.

**Requirement Satisfaction**

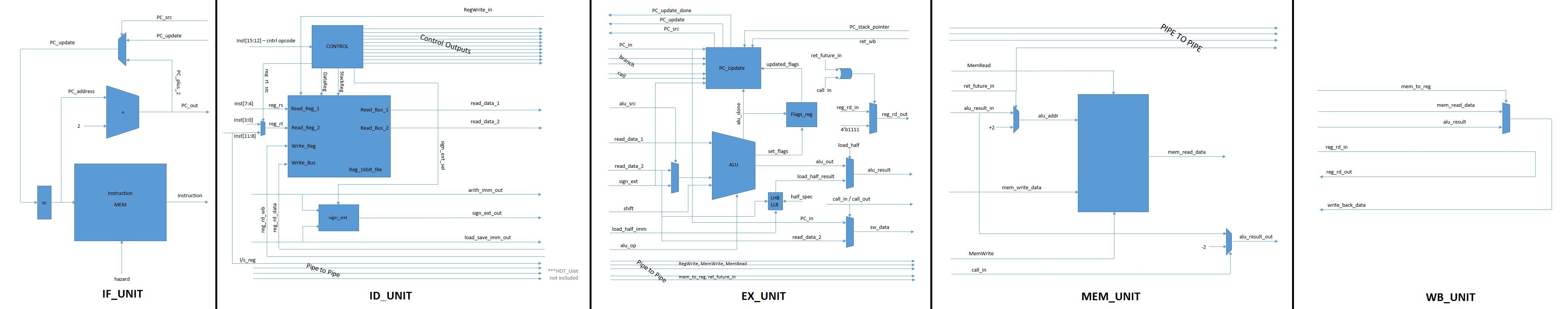
Our WISC-S15 architecture meets nearly all requirements of the intermediate project specification; these small exceptions are as follows:

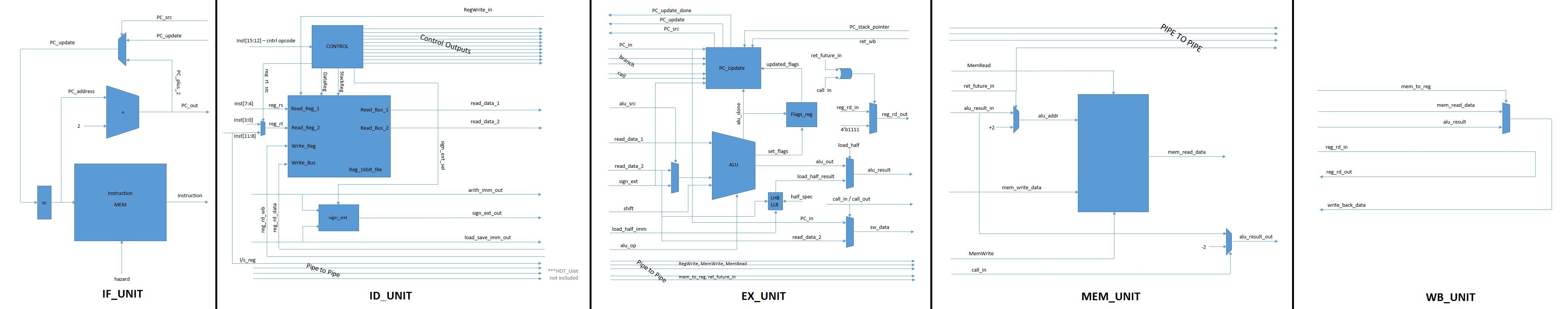
1. LW operation — this operation *is functional*; our team has verified this by performing a SW operation, followed by a delay, followed by a successful LW operation. This is listed as an exception because our team saw issues with the sign\_extension signal arriving too early, causing the LW to fail sometimes.
2. BRANCH operation – this operation contains a small bug, but works *most of the time*. Additionally, we did not have time to implement the delayed branching aspect; this will be included in our final design.

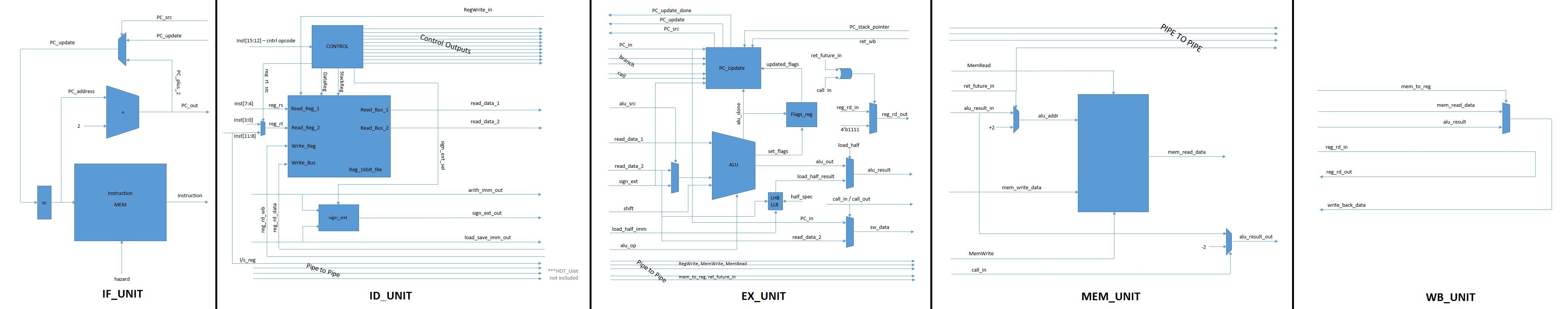
*VERILOG CODE*

The Verilog code and test benches for our WISC-S15 architecture has been included within this zip file, under the “code” folder.

*ARCHITECTURE DIAGRAM*

For your convenience, we have included a five-stage diagram illustrating the five pipelined sections of our architecture. The pipeline registers and Hazard Detection Unit have been omitted due to spatial constraints and for increased readability:



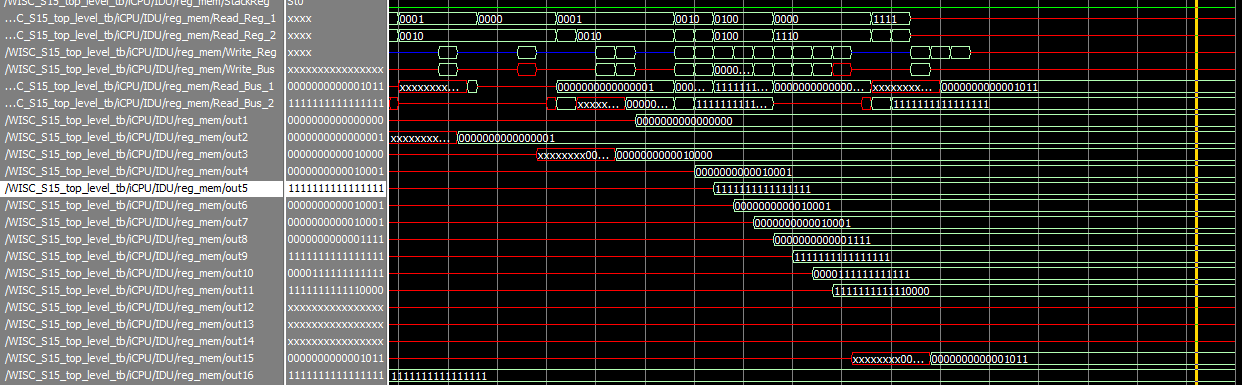


*AREA REPORT*

Our architecture Verilog code was synthesized using the script provided; the following Area Report was generated:

*SIMULATION RESULTS*

Our architecture Verilog code was simulated using the test programs provided. The resulting information can be found under “tests\test\_results” folder. The following figure is one of the simulation results, exemplifying each instruction output corresponding to the numbered register:



*SUMMARY*

Student Names: Graham Nygard, Robert (Bob) Wagner

Area of Design (except memory, cache):

|  |  |  |
| --- | --- | --- |
| Name of Program | Execution Time (pre-synthesis) | Execution Time (post-synthesis) |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

*Please note, the professor explicitly specified that we were not responsible for the post-synthesis results in this intermediate project report.*