Cameron Johnson

Levi Amen

Will Preachuk

Processor Design Specifications

In the course of designing our processor, multiple factors needed to be considered in order to both properly plan out a work schedule as well as allowing for the best possible processor being created.

At its most basic, our processor is designed to both handle and calculate 32 bit values. Inside of our processor is a register file of 32 registers capable of holding 32 bit values. We chose to make a 32 bit processor due to the extent and capability of expansion that a 32 bit system allows for. A 32 bit instruction is capable of handling more operation codes and thus is able to process multiple operations. It also allows for more memory as memory addresses may be larger. Another advantage of the 32 bit concept is that it allows the calculation of much larger numbers with twice as many significant figures as a 16 bit architecture.

Each operation inside of our processor requires an operation code, held in a specific place so as to easily be read by the processor. The length of our operation code is 5 bits wide allowing for 32 possible operation, this design choice was the best way to allow for the minimum amount of operations required by the project as well as room for improvement and development of extra features. On top of this, a 5 bit op code size is relatively small, allowing more space to be allocated for immediate values and other register information, as well as extended op codes for use in the ALU.

Our ALU takes an op code of two bits. This forced an interesting design decision on our group. We choose that every operation that would be calculated by the ALU would use the same OP code, and simply use the Extended OP code field to specify which instruction the ALU was to return. Thus, our Add, Sub, AND, OR and XOR all have the same op code. On top of this, we specified our addi instruction to have a very similar op code. While Add used 00000, addi used 01000. This would specify that it should go to the ALU, but that the immediate value would be used instead of a register. This was done for consistency's sake.

Finally the proposed syntax for our assembly language is highly inspired by the NIOS II assembly language, due to familiarity that our group has with the language.