Overview:
In conjunction with other groups we will use the VHSIC Hardware Description Language (VHDL) to design and implement a functional Reduced Instruction Set Computer (RISC) Processor. We will implement the processor using a Field Programmable Gate Array (FPGA), allowing us to develop a processor that can be easily and quickly re-configured for specific applications. Our VHDL design and simulation will be done using Mentor Graphics ModelSim software. A XESS Development Board with one of the XILINX XC4000e family of FPGA's will be used for the implementation. Our group will focus on the Control Unit of the processor. This section of the processor fetches the instruction from memory, generates the appropriate control signals for the rest of the processor, and handles program branches.
RISC Background:
The RISC Processor is based upon the idea that a small, basic instruction set in conjunction with a “smart” compiler can deliver superior performance over a Complex Instruction Set Computer (CISC) with a large number of specialized instructions. The simplicity of the operations performed ideally allows every instruction to be completed in one processor cycle.

The basic datapath of a RISC processor is shown below in Figure 1. The Instruction Fetch loads the instruction pointed to by the program counter (PC) from processor memory. The Instruction Decoder then generates the appropriate control signals for the Execute unit, which performs the desired function (arithmetic, logic, etc.) on the data. The Writeback unit then updates the memory with the new values.

Figure 1: RISC Processor Datapath

Functionality:
Our task is to design the control section of the processor, the Instruction Fetch and Instruction Decode units. The Instruction Fetch unit is responsible for determining the location of the next instruction based on the value of the program counter (PC) and branch condition signals generated by the Execute Unit.

The Instruction Decode unit takes this instruction and generates control signals to be sent to the other units. After generating these signals the unit loads appropriate values or immediate data into processor registers used by the Execute Unit.

Depending on the type, the instruction could contain immediate data, the address of processor registers, or a target offset in the case of a branch instruction. Our decoder will extract these values and transfer them to the appropriate processor registers and units.
**Inputs:**
- **BR (Branch):** Contains the target offset address for a branch
- **PC (Program Counter):** Points to the next instruction to be executed
- **IR (Instruction Register):** The instruction to be executed

**Outputs:**
- **MW (Memory Write):** Allows data memory write
- **RW (Read-after-write):** Allows read of processor register after writeback
- **DA (Data Address):** Address of data to read for read-after-write
- **DS (Data Select):** Select Function output, data output, or target offset
- **BR (Branch):** Contains the target offset address for a branch
- **FS (Function Select Bus):** Word which contains the op-code
- **A (A Data Bus):** Word which contains data for the A input of Execute unit
- **B (B Data Bus):** Word which contains data for the B input of Execute unit

---

**Figure 2: Function Block Diagram**