Computer-architecture Projects:

1. Write a MIPS assembler, whose input is a MIPS assembly code, and the output is the instruction code in binary for each instruction. Test the assembler on a matrix multiplication program. You can write this in C/python/verilog
2. Design a pipelined MIPS processor which will take as input- the instruction codes for an N factorial program and produces the necessary output. Implement stalls/NOPs when there are dependencies. In how many clock cycles does the output of the program appear?
3. Design a non-pipelined MIPS processor which will take as input- the instruction codes for an N factorial program and produces the necessary output. In how many clock cycles does the output of the program appear?
4. In Verilog - design a cache simulator for a direct-mapped cache to hold 256 lines, 16 words per line (word = 32 bits). Assume a 32 bit address. Implement a main memory. Implement an LRU scheme for replacement. Use the memory trace files at this location as input. The trace file will specify all the data memory accesses that occur in a certain program. Each line in the trace file will specify a new memory reference and has the following fields:
    a. Access Type: A single character indicating whether the access is a load ('l') or a store ('s').
    b. Address: A 32-bit integer (in unsigned hexadecimal format) specifying the memory address that is being accessed.
    c. Instructions since last memory access: Indicates the number of instructions of any type that executed between since the last memory access (i.e. the one on the previous line in the trace). For example, if the 5th and 10th instructions in the program's execution are loads, and there are no memory operations between them, then the trace line for with the second load has "4" for this field.

    For different sizes of the block, for eg- 1 word per line, 2 words per line etc upto 16, what are the hit/rates ? In other words, plot the miss/hit rates for varying word size per block.
5. In this project, implement a cache simulator for an N-way set associative cache with one word per block. Implement a main memory. Choose the number of sets such that the size of the cache is 32MBytes. You can use the same trace files as in Project 4 above. How would the hit/miss rate change with varying associativity: 2- way, 4-way, 8-way. Use LRU for replacement
6. Design a 2-level cache hierarchy with memory. L1 2-way 32kB, L2 4-way 64MB. Use LRU at both levels. Demonstrate cache reads, hit/ miss ratio. Use the trace files from Project 4.
7. Design a 2-level cache hierarchy. L1 2-way 32kB, L2 4-way 64MB. Use LRU at both levels.128MB main memory is also in place. Use trace files from Project 4 for memory addresses. Demonstrate write through operation and compare it with write-back operation.

8. Design a 2-level cache hierarchy. L1 2-way 32kB, L2 4-way 64MB. Demonstrate cache reads. Use 3 different replacement policies (L1 and L2 having same replacement policy) and compare hit rate and miss rate for all 3 cases. Use trace files from Project 4 for memory addresses.
9. If you want to implement any other project, write up the project proposal and email me.