Department of Electrical and Computer Engineering
University of Texas at Austin

ECE 382N, Spring 2000
Y. N. Patt, M. D. Brown
Homework #2, Due: February 18, 2000.

Part 1. Describe, using register transfer notation, the functionality of the following subset of the Intel x86 architecture:

instructions: ADD, NOT, MOV(data), INC, JMP.
addressing modes: (using r/m): immediate, register, base+displacement.
data types: doubleword.

Part 2. Design the data path and an accompanying state diagram to implement the subset of the x86 architecture described in Part 1 above.

Your state diagram should show all the relevant states. That is, pay careful attention to each phase of the instruction cycle, and ALL the processing that must go on to implement these instructions.

For this assignment, we will not be concerned about interrupts or checking via "machine checks" or traps/faults for illegal situations. (That comes later!)

The data path need not show control signals, just information paths.

Part 3. You can use whatever library parts that are available for this assignment. These include a 4-bit ALU slice. Please hand in all circuit diagrams for your design. You may create dummy modules in Verilog to "generate" the control signals, such as the one shown below:

```
module ALU_control (opcode, alu_control_signals);
input  [N:0] opcode;
output [M:0] alu_control_signals;
assign alu_control_signals = 0;
endmodule
```

For this assignment, no simulation is necessary. However, your Verilog code should be able to compile correctly.

You do not have to turn in a copy of your Verilog code. Please put a copy of your code in a directory called **hw2** within your class directory. When you continue working on your code for assignment 3, use a separate directory.

Note: Even though you will not be turning in your verilog code and simulation runs, it is to your advantage to completely test it now. It will save you time on the following assignments and on the project if you can start with a fully-debugged datapath.

Part 4. For the design of part 2, select an instance of each of the five instructions, choosing appropriate addressing modes for each. Calculate the number of cycles required to execute each of those five instructions. Start counting cycles at the beginning of an instruction's fetch cycle and end with that instruction's completion (for example, destination write). For purposes of this assignment only, assume 10 nsec cycle time, single-cycle cache access, 100 nsec memory access time, data cache hit ratio of 0.80, instruction cache hit ratio of 0.95. and no page faults. Please show your calculations.

Note: The actual number of cycles required to execute your program will be substantially fewer, since you will be able to overlap instruction execution. (That's what pipelining has been all about for more than 30 years). But, for this assignment only, we will examine the execution times of each instruction individually.