

ECEN 5253

Fall 2006

Design Project 4

For this assignment, you will design a simplified 32 bit pipelined CPU as shown in fig. 1

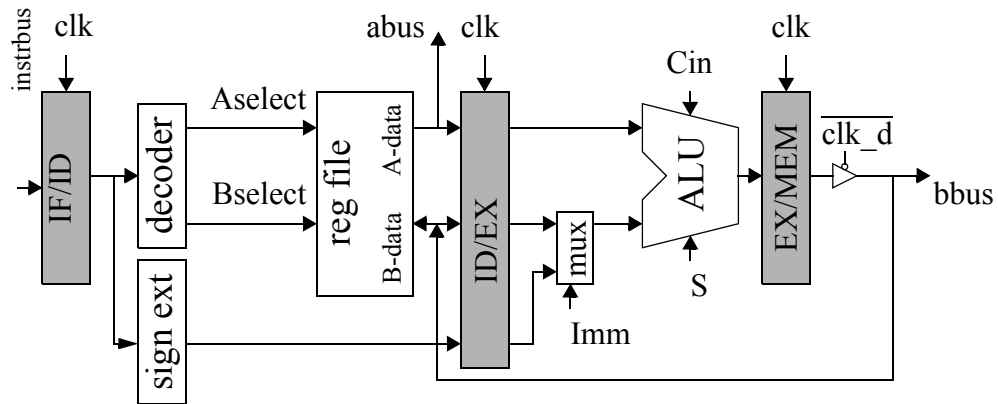


FIGURE 1. Pipeline Block Diagram
so that the following subset of instructions is implemented.

TABLE 1. Subset of instruction codes to be implemented.

Code	Func	Name	Format
000011		ADDI	I
000010		SUBI	I
000001		XORI	I
001111		ANDI	I
001100		ORI	I
000000	000011	ADD	R
000000	000010	SUB	R
000000	000001	XOR	R
000000	000111	AND	R
000000	000100	OR	R

The instruction formats should be as in Patterson and Hennessey, inside front cover, except that only the I and R formats are implemented and the logic operations are slightly different. All other op-codes should be treated as NOPs.

Part a: Due Fri., Oct. 27, 1700 hrs.

Design the controller for the pipelined CPU. The control points are the Aselect and Bselect inputs to the register file (leave the rd and wr controls as in project 3), the Imm input to the MUX and the S and Cin inputs of the ALU as shown in fig. 1. The contents of the instruction register must be decoded to provide all of the control inputs. Binary decoders can be used to produce the Aselect and Bselect inputs for the register file from the operand fields in the instruction register. As discussed in the lecture notes on pipeline control, the Bselect comes from a different register field for reading the B source than for writing the destination. Write the logic equations for the decode logic so that the control lines are correct for each instruction. It is probably a good idea to assign a default inactive value for each control line since you do not want to leave control lines undefined at any time. Make sure that the control lines go through the appropriate pipeline registers before connecting to the control points. The controller design might be similar to fig. 2.

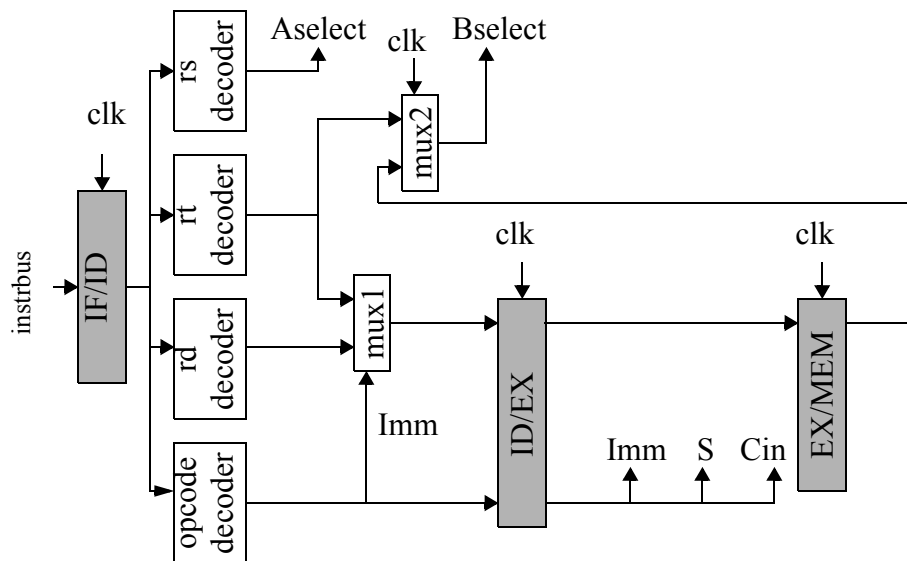


FIGURE 2. Controller Block Diagram

The mux1 is needed because the destination register is in the rd field for R format, but in the rt field for I format. The mux1 control is the same Imm as in the data path. The mux2 should select rt when clk is low for reading and the destination register when clk is high for writing.

To facilitate testing of your design, the IF/ID register inputs will be connected to the instruction bus which will be made available as an external terminal. Your design should have the terminals shown in figure 3.

You must turn in to the automatic grader a list of all Verilog files in a file named controller.ver and your high level Verilog file must be named controller.v. Other Verilog files can be named as you wish. When using the automatic grader, be sure to use the correct class and assignment name (spelling and case is important!).

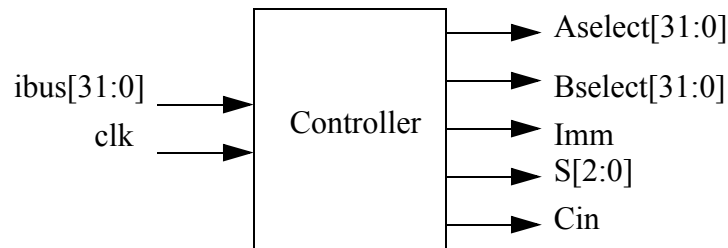


FIGURE 3. Controller Terminals

class: ecen5253
assignment: proj4a

Part b: Due Fri., Nov. 10, 1700 hrs.

Connect the controller to the data path hardware in fig. 1. The data path is similar to the register file and ALU which has already been put together in project 3. You will need to add:

1. the sign extension logic which extends the lower order 16-bits of the instruction to 32 bits, and
2. a MUX on the ALU B-input such that the sign extension output is selected for I format.

To facilitate testing of your design, the IF/ID register inputs will be connected to the instruction bus which will be made available as an external terminal. Your design should have the terminals shown in figure 3.

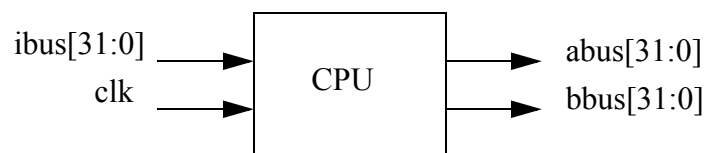


FIGURE 4. Controller Terminals

Each instruction will stay on the ibus for one clock cycle. It is acceptable for all of the registers in the register file to remain undefined until something is clocked into them.

You must turn in to the automatic grader a list of all Verilog files in a file named `cpu4.ver` and your high level Verilog file must be named `cpu4.v`. Other Verilog files can be named as you wish. When using the automatic grader, be sure to use the correct class and assignment name (spelling and case is important!).

class: ecen5253
assignment: proj4b