LAB 4: Support Material: CPU

Description

Pipelining

Pipelining is an implementation technique whereby multiple instructions are overlapped in execution. In this lab, we provide a three-stage pipelined CPU (Fig. 1 and Fig. 2) which is based on RISC instruction set. Every instruction can be implemented at most 3 clock cycles. The 3 clock cycles are as follows and the example figure is shown in Fig. 2.

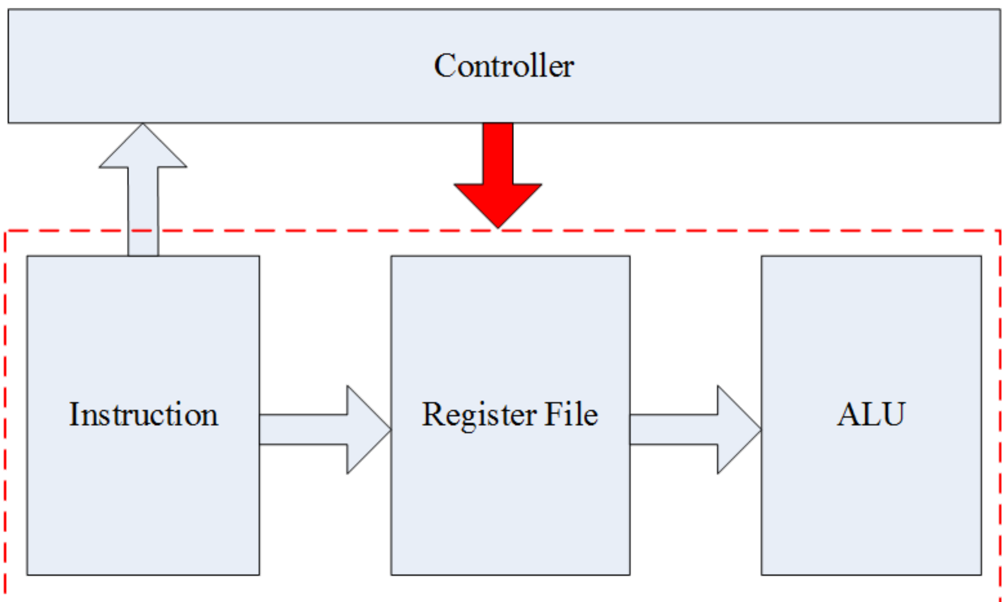

Figure 1 Block Diagram of the System

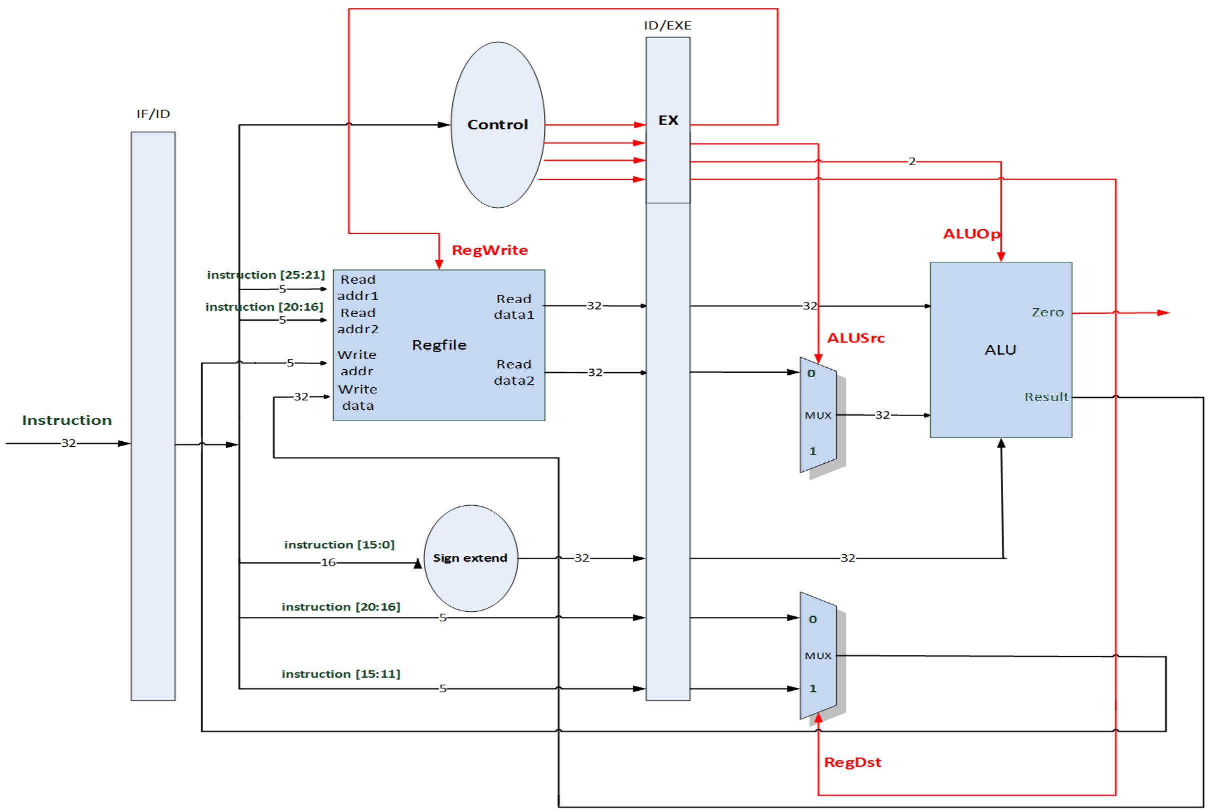

Figure 2 Block Diagram of 3-stage Pipelined Datapath with Control

1. Instruction fetch cycle (IF):
   Fetch the current instruction from the test bench to make the CPU work.
2. Instruction decode/register fetch cycle (ID):
   Decode the instruction and read the registers corresponding to register source specifies from the register file.
3. Execution/effective address cycle (EXE):
   The ALU operates on the operands prepared in the prior cycle, performing one of three functions depending on the instruction type.
   - Memory reference:
     The ALU adds the base register and the offset to form the effective address.
   - Register-Register ALU instruction:
     The ALU performs the operation specified by the ALU opcode on the values read from the register file.
   - Register-Immediate ALU instruction:
     The ALU performs the operation specified by the ALU opcode on the first value read from the register file and the sign-extended immediate.

Figure 3 Simple RISC Pipeline

Instruction	t	t + 1	t + 2	t + 3	t + 4	t + 5

Instruction	t	t + 1	t + 2	t + 3	t + 4	t + 5
i	IF	ID	EXE			
i + 1		IF	ID	EXE		
i + 2			IF	ID	EXE	
i + 3				IF	ID	EXE

Instruction

In this simple 3-stage pipelined CPU, we use a RISC (reduced instruction set computer) architecture to illustrate the basic concepts, although nearly all the ideas we introduce in this lab are applicable to other processors. The instruction format is as follows.

Table 1 Supported Instruction Set

Type	Description
Instruction formats	All 32 bits wide (one word)
Opcode	Operation code
Rs	First input source general purpose register
Rt	Second input source general purpose register
Rd	Output destination general purpose register
Shamt	Shift amount
Function	Function mode

- R-type format:

Opcode	Rs	Rt	Rd	Shamt	Function
6-bit	5-bit	5-bit	5-bit	5-bit	6-bit

Instruction	Opcode	Function	Operation	PC
NOP	000000	000000	N/A	PC = PC + 4

Instruction	Opcode	Function	Operation	PC
ADD	000000	100000	Rd = Rs + Rt;	PC = PC + 4
SUB	000000	100010	Rd = Rs – Rt;	PC = PC + 4
AND	000000	100100	Rd = Rs and Rt;	PC = PC + 4
OR	000000	100101	Rd = Rs or Rt;	PC = PC + 4
XOR	000000	101000	Rd = Rs xor Rt;	PC = PC + 4
SLT	000000	101010	If Rs < Rt, Rd = 1; else Rd = 0;	PC = PC + 4
SLL	000000	000011	Rd = Rt <<< shamt;	PC = PC + 4
SRL	000000	000010	Rd = Rt >>> shamt;	PC = PC + 4

- I-type format:

Opcode	Rs	Rt	Signed Immediate Value
6-bit	5-bit	5-bit	16-bit

Instruction	Opcode	Operation	PC
ADDI	001000	Rt = Rs + immd;	PC = PC + 4
SET	000001	Rt = immd;	PC = PC + 4

Module

The simple 3-stage pipelined CPU consists of Controller, Register file and ALU. The description of these blocks is as follows.

- Controller:
  Receive input instruction and according to the different instructions to output the corresponding control signals. (RegWrite, RegDst, ALUSrc, and ALUOp are generated control signals)
- Register file:
  Input the address of registers, which we want to use, to Read addr1 and Read addr2, then the Read data1 and Read data2 will output the corresponding address of register data. Likewise, Write addr is the address of register we want to write the data into register file.

- ALU:
  Depend on the control signal from Controller to do the corresponding operation.

Table 2 Control Signals

Instruction	Opcode	RegWrite	RegDst	ALUSrc	ALUOp
R-type	000000	1	1	0	10
ADDI	001000	1	0	1	00
SET	000001	1	0	1	00

Table 3 ALU Decoder Truth Table

ALUOp	Function	Operation
00	X (don't care)	ADD
01	X (don't care)	SUB
10	000000	NOP
10	100000	ADD
10	100010	SUB
10	100100	AND
10	100101	OR
10	101000	XOR
10	101010	SLT
10	000011	SLL
10	000010	SRL