CSE-306

Assignment on 8-bit MIPS Pipelined Execution

Submitted by:

Group: 1

Section: B-2

Members:

Nahian Salsabil - 1705091

Farhana Khan - 1705100

Rittik Basak Utsha - 1705105

Muhtasim Noor - 1705108

Introduction

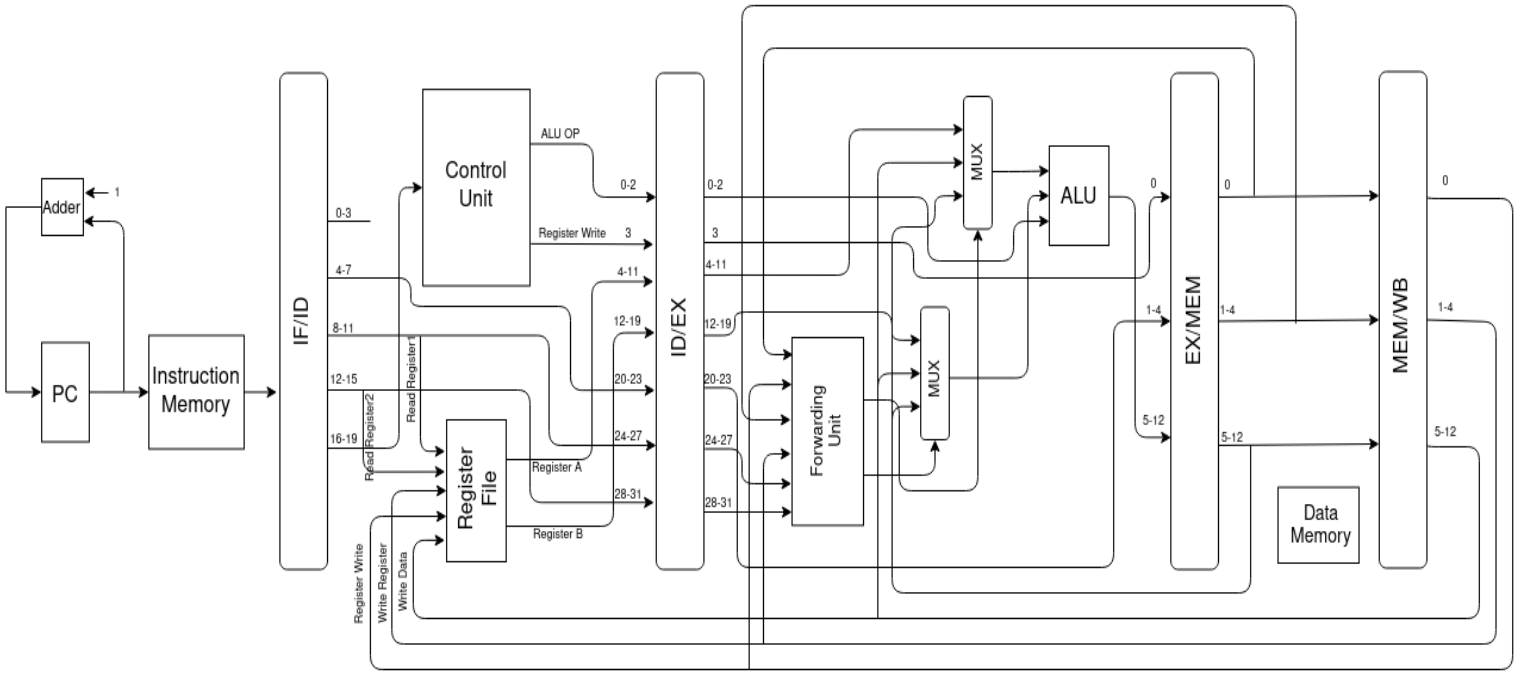
In this assignment, our goal was to design an 8-bit processor that supports pipelined data-path for a subset of MIPS instruction set. We had to only consider four R-type instructions. Each instruction had been divided into five stages: instruction fetch (IF), instruction decode (ID), execution and address calculation (EX), data memory access (MEM), and write back (WB).

The length of the clock cycle equals the maximum time to execute any single stage. Therefore, each instruction takes up to *five* clock cycles to be executed. The main components of the processor are as follows: instruction memory, data memory, register file, ALU, control unit, four pipeline registers, and a forwarding unit. Some additional components were used in designing the processor.

In this assignment, we only considered data hazards that can be resolved by forwarding. Particularly, we took care of the following hazards:

- **EX Hazard:** occurs when the dependent instruct is in the EX-stage and the prior instruction is in MEM stage.
- **MEM Hazard:** occurs when the dependent instruction is in the EX-stage and the prior instruction is in WB stage.
- **Double Data Hazard:** occurs when the dependent instruction is in EX stage and it depends on two prior instructions, one of which, is in MEM stage, and the other is in WB stage.

Complete Block diagram of pipelined datapath



Block diagrams and size of pipeline registers

IF/ID Register:

size: 20 bits

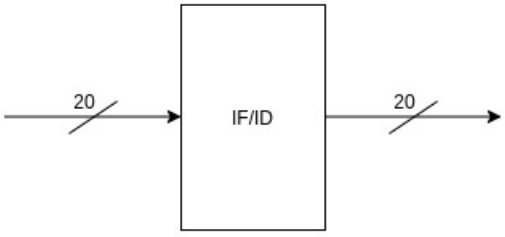


Figure: IF/ID Register

ID/EX Register:

size: 32 bits

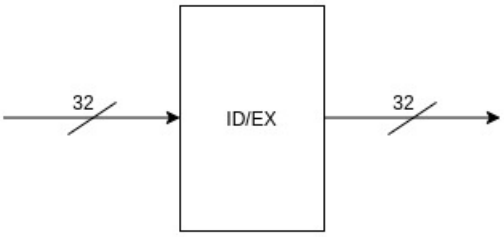


Figure: ID/EX Register

EX/MEM Register:

size: 13 bits

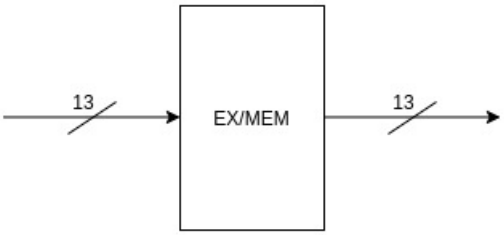


Figure: EX/MEM Register

MEM/WB Register:

size: 13 bits

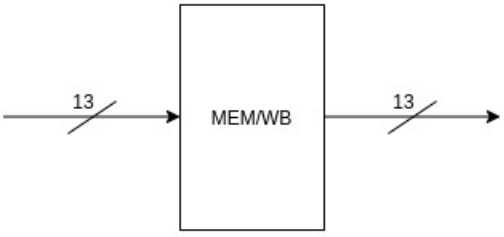


Figure: MEM/WB Register

Mechanism and block diagram of Forwarding Unit

- The forwarding unit takes the following values as input
  → Read Register 1 from ID/EX register

  → Read Register 2 from ID/EX register

  → Write Register from EX/MEM register

  → registerWrite signal from EX/MEM register

  → Write Register from MEM/WB register

  → registerWrite signal from MEM/WB register

  Where,

Read Register 1	holds the value indicating the first source register
Read Register 2	holds the value indicating the second source register
Write Register	holds the value indication the destination register
registerWrite	signal from the control unit, indicates the instruction requires to update the value of Write Register

- If the read register from ID/EX is the zero register, the corresponding register data is not forwarded. Therefore, ALU gets data provided by the register unit.

- If the read register from ID/EX is equal to the write register of the EX/MEM register and the registerWrite signal is set, data is forwarded from EX/MEM unit to the corresponding register.

- If the read register from ID/EX is equal to the write register of the MEM/WB register and the registerWrite signal is set, data is forwarded from MEM/WB unit to the corresponding register.

- If a certain register fulfills the aforementioned conditions for both EX/MEM and MEM/WB units, data is forwarded from EX/MEM unit.

- In all other cases data is not forwarded. Therefore, ALU gets data provided by the register unit.

- The forwarding unit generates 2 signals ForwadA and ForwardB, each of two bits length, according to the following table.

Table 1: explanation of Forwarding unit MUX control outputs

MUX control	source	Explanation
ForwardA = 00	ID/EX	the first ALU operand (A) comes from the register file
ForwardA = 10	EX/MEM	the first ALU operand (A) is forwarded from prior ALU result
ForwardA = 01	MEM/WB	the first ALU operand (A) is forwarded from two clock pulse prior ALU result
ForwardB = 00	ID/EX	the first ALU operand (B) comes from the register file
ForwardB = 10	EX/MEM	the first ALU operand (B) is forwarded from prior ALU result
ForwardB = 01	MEM/WB	the first ALU operand (B) is forwarded from two clock pulse prior ALU result

The required truth tables and functions for generating ForwardA and ForwardB signals are as follows:

Table 2.1: table for determining if data is required to be forwarded from EX/MEM unit

EX/MEM.destination == ID/EX.source	EX/MEM.regWrite	ID/EX.source == 0	Xsource
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

Table 2.2: table for determining if data is required to be forwarded from MEM/WB unit

MEM/WB.destination == ID/EX.source	MEM/WB.regWrite	ID/EX.source == 0	Msource
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

Table 2.3: table for determining MUX control, ForwardA

XA	MA	ForwardA1	ForwardA0
0	0	0	0
0	1	0	1
1	0	1	0
1	1	1	0

Table 2.4: table for determining MUX control, ForwardB

XB	MB	ForwardB1	ForwardB0
0	0	0	0
0	1	0	1
1	0	1	0
1	1	1	0

Functions:

Xsource = **(** EX/MEM.destination == ID/EX.source **).(** EX/MEM.regWrite **).(** ID/EX.source == 0 **)**

Msource = **(** MEM/WB.destination == ID/EX.source **).(** MEM/WB.regWrite **).(** ID/EX.source == 0 **)**

Forward0 = X′M

Forward1 = X

Forwarding Unit:

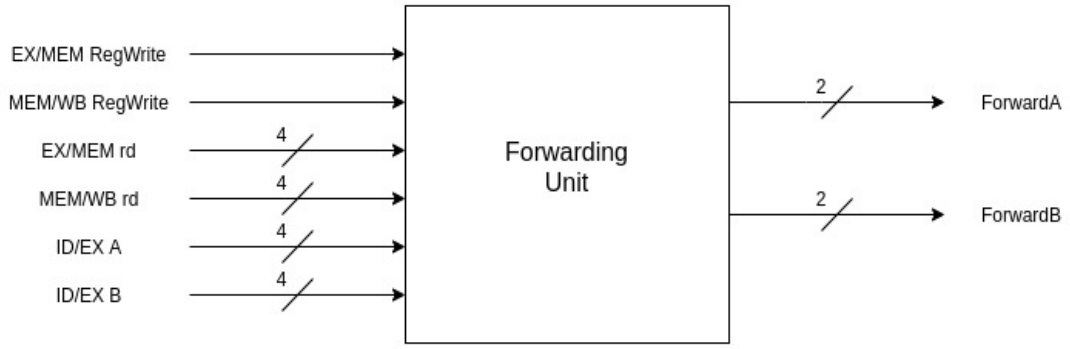


Figure: Forwarding Unit

Discussion

The following were taken into consideration:

- A program has been written in C++ language to convert the given assembly code into MIPS machine code. This can be loaded automatically into the circuit simulator.
- The control unit has been micro-programmed. Control signals associated with the operations is stored in a separate ROM as Control Words.
- All clocks required in the circuit is provided from a single clock source. Each instruction has been fetched by a single clock cycle.
- Two separate memory units are used to implement the instruction memory (ROM) and data memory (RAM).
- While designing, minimizing IC count and following the MIPS design principles were prioritized.