

Multi-Cycle MIPS Implementation

Tom Kelliher, CS 240

Apr. 1, 2002

1 Administrivia

Announcements

Assignment

Read 5.4. Be ready to work control sequencing exercises in class.

From Last Time

Comparison between single- and multi-cycle implementations.

Outline

1. Introduction.
2. Components added to the datapath.
3. The complete datapath and its control signals.

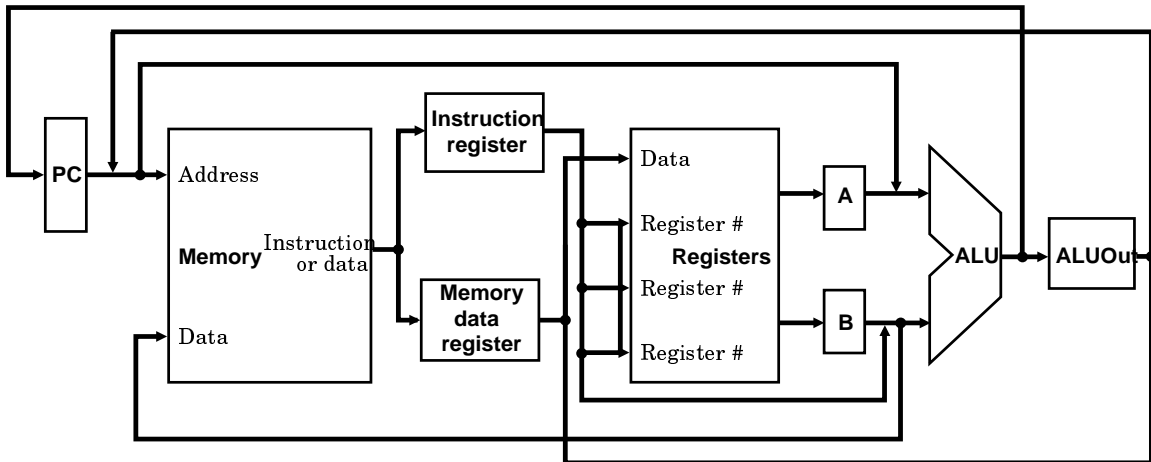
Coming Up

Controlling the multi-cycle implementation.

2 Introduction

1. Instruction cycle broken into a number of clock cycles.
2. Functional units can be shared: Use ALU for incrementing PC. Hardware savings.
3. Instructions can take differing numbers of clock cycles.

High-level block diagram:



Features relative to single cycle implementation:

1. Single, unified memory.
2. Eliminated two adders.
3. Registers added between major functional to hold values for next clock cycle. *Are they architectural registers?*

3 Additional Registers

Two uses for additional registers:

1. Fitting the clock cycle.

2. Holding values needed during a later clock cycle of *this* instruction.

Example: computing and holding branch target address *before* the instruction is decoded.

Determining where to place registers for timing purposes:

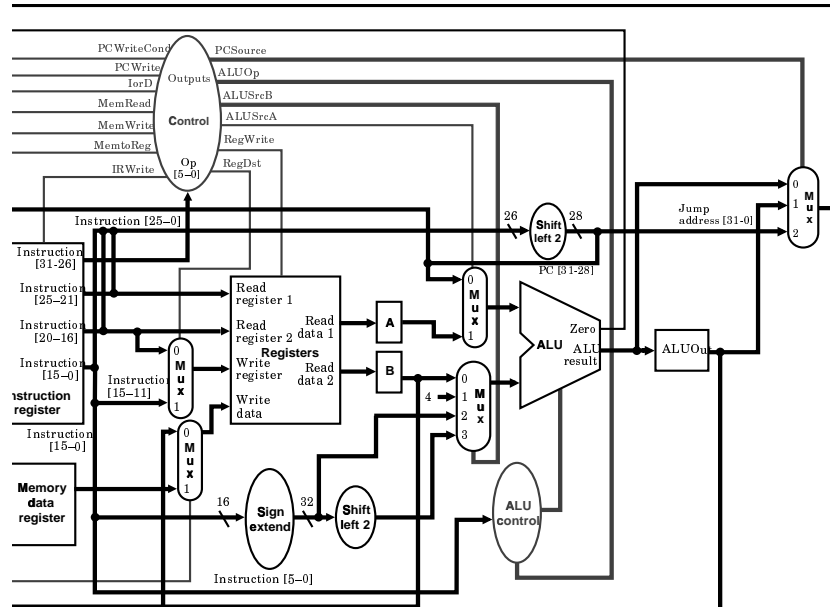
1. All combinational paths must fit into one clock cycle.
2. Assume that a clock cycle can accommodate at most one of the following:
 - (a) Memory access.
 - (b) Register file access.
 - (c) ALU operation.
3. The added registers:
 - (a) Instruction register (IR): Holds the currently executing instruction.
 - (b) Memory data register (MDR): Holds the memory word from a `lw`.
 - (c) Register file read registers (A and B): Buffers read data from register file.
 - (d) ALU out register (ALUout): Buffers output from ALU.

4 Additional ALU Inputs

Needed because we eliminated the adders. We'll need to add them back later.

1. PC to upper input.
2. Constant 4 and sign-extended, shifted immediate field to lower input.

5 The Complete Datapath



Control signals:

1. RegDst, RegWrite.
2. ALUSrcA: Choose between PC and Rs.
3. MemRead, MemWrite, MemtoReg.
4. IorD: Choose between PC and ALUOut for memory address.
5. PCWrite: Load a new value into PC.
6. PcWriteCond: Load a new value into PC if zero is active.
7. ALUOp.
8. ALUSrcB: Choose between Rt/Rd, 4, sign-extended immediate, sign-extended shifted immediate.
9. PCSource: Choose between PC + 4, ALUOut (branch target address), jump address