

David M. Koppelman

1.1 Introduction

An important part of the course and a big chunk of midterm- and final-exam credit is on the statically scheduled MIPS implementations. Essentially two kinds of questions are asked about these: *How will it run this program?* and *How can it be modified to execute this new instruction?*

This study guide lists questions of each type to study, along with hints on how to solve them. The questions are grouped by type, from easiest type to hardest type. Once problems in one group becomes easy (or at least doable) move on to the next group.

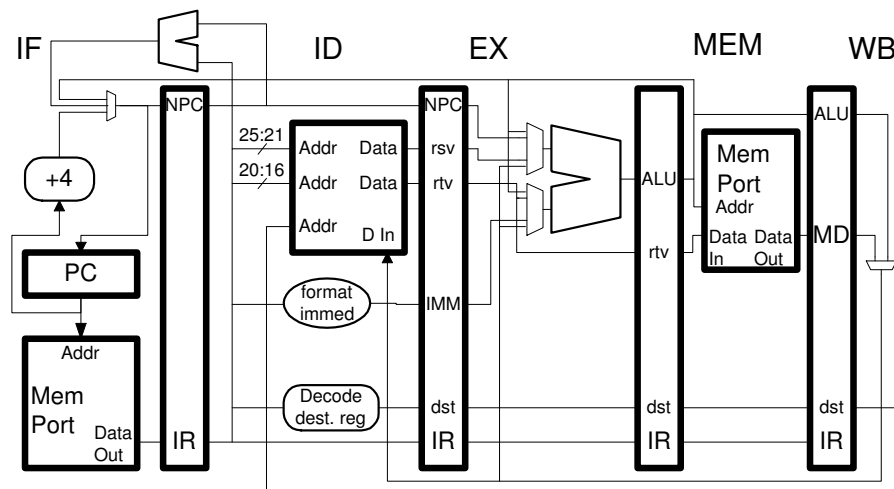
1.1.1 General Study Hints

In the hardware diagrams, such as the one illustrated below, every wire has a purpose. Before attempting the problems below look over the diagram and if necessary find out the purpose of any wires you are not familiar with. The same holds for logic, such as the ALU, format immediate unit, etc.

You should be able to determine what value will be on a wire at each cycle when a given program runs (see the *PED and Values problems*) and you should be able to write a program that will set the wire to a particular value at a particular cycle (see the *Write a program consistent...* problems).

The following have been difficult for some:

- Stores (**sb**, **sh**, **sw**: those stores). Understand that the value being stored goes through the **rtv** latch between EX and MEM.
- Branch conditions. The comparison units for branches are not often shown, but they do read registers and so are the cause of stalls. They are usually in the ID stage.
- Jump and link (**jal**, **jalr**). Don't forget about saving the return address. The return address is the value of the NPC latch, it usually makes its way into the ALU in the EX stage.
- Control transfers (branches and jumps). You must know the exact cycle the target will be fetched.



1.2 Problems, By Type

Problem types are described below in order of increasing difficulty. The difficulty is shown in the title.

1.2.1 Easy: PED and CPI

Given a program and hardware, show a pipeline execution diagram (PED). In some variations, determine the CPI for a large number of iterations. Most PED credit will be given for inserting the correct stalls and fetching the CTI targets at the right cycle. The PED problems are usually easy, the CPI questions can get tricky.

- Be sure to check the program for dependencies.
- For each dependency determine whether a stall is necessary. (Start by drawing a PED without a stall, then check to see if the value can pass from the instruction writing the register to the instruction reading it. If not, insert stalls.)
- For CPI, be sure to correctly identify when iterations start (on the IF of the first instruction) and use a repeating pattern of iterations for determining the cycle count.

Basic problems: Fall 1999 Homework 2 Problem 3, Spring 2002 Homework 2 Problem 2, Spring 2004 Final Exam Problem 2(a)

A more difficult problem of this type: Fall 2000 Midterm Problem 3

1.2.2 Easy and medium: PED and Values

Given a program and hardware, show the values that will be present on selected wires at the indicated cycles (either specified in a table or labeled on the diagram). This is a more detailed version of the PED problem.

Basic Problems: Fall 2005 Homework 4 Problem 1 (for branches), Spring 1999 Homework 3 Problem 2, Fall 1999 Homework 2 Problem 2, Spring 2000 Homework 2 Problem 1, Fall 2005 Final Exam Problem 2 (includes some non-statically scheduled parts).

More Challenging: Fall 2005 Final Exam Problem 1 (hardware uses a loop-count register, something not usually shown).

1.2.3 Easy: Add Bypass and Other Paths

Add bypass and other connections so the program executes as shown. In most of these problems a PED is shown that could not have executed on the illustrated hardware because the hardware lacks certain bypass and other connections. The problem is to add exactly the paths that are needed. ("Exactly" means points will be deducted [sorry] for adding bypass paths that are not needed.)

Problems:

Spring 2002 Homework 2 Problem 3, Fall 2005 Homework 4 Problem 2 (for branches), Spring 2000 Homework 2 Problem 2, Fall 2000 Midterm Problem 1, Fall 2001 Midterm Problem 1, Fall 2001 Final Exam 4e, Fall 2000 Homework 3 Problem 3, Spring 2004 Final Exam Problem 2(b)

1.2.4 Medium: Write a program consistent with values on labeled wires.

The values on certain wires at certain cycles are shown. Write a program that could have produced those values. This problem type is sort of like a puzzle, reconstructing a program using a few clues.

Problems,

Spring 2005 Midterm Problem 1 (easy), Spring 2005 Midterm Problem 2 (includes floating-point pipelines), Fall 2004 Midterm Problem 1, Fall 2003 Midterm Two Problem 1, Spring 2003 Midterm Problem 1, Fall 2002 Final Exam Problem 1, Spring 2002 Midterm Problem 1, Spring 2001 Midterm Problem 1

1.2.5 Medium: Control Logic

Design the control logic for a multiplexor. These are the easier hardware design problems. To solve it, determine when the multiplexor inputs are used then design logic to detect those situations. Be aware that the logic is usually in the decode stage but that the signal is used in a later stage.

Problems:

Fall 2001 Midterm Problem 2, Spring 2001 Midterm Problem 3a, Fall 2003 Final Exam Problem 3(c), Spring 2004 Final Exam Problem 2(c,d)

1.2.6 Hard: New Instructions

Modify the illustrated hardware so that it implements a new instruction. These are the most time-consuming and so usually appear on the final exam.

To solve this type of problem, first understand what the instruction is supposed to do. Then, write a short program using the instruction and determine the PED of its execution. (Sometimes the PED will be given.) Determine what values the new instruction will have to read and write. For example, it might require reading and writing a new register (separate from the general-purpose register file). Determine if extra arithmetic units or other computation hardware is needed. Finally, add control signals for the hardware (if requested).

Pay attention to how the new instructions are affected by dependencies.

Problems:

Spring 2005 Final Exam Problem 1 (multiply-add instruction), Fall 2004 Final Exam Problem 2 (memory-to-memory arithmetic instructions), Fall 2003 Midterm Two Problem 2 (control logic for post-increment add datapath), Spring 2002 Final Exam Problem 2 parts a,b, and c;

Fall 2002 Midterm Problem 1, Fall 1999 Homework 2 Problem 5, Fall 2000 Homework 3 Problem 1, Fall 2000 Midterm Problem 2, Fall 2000 Final Exam, Spring 2000 Final Exam Problem 1, Spring 2001 Final Exam Problem 1