

Problems 2

Problem 1

- ❑ A datapath that executes all instructions in a single clock cycle was presented in Section 5.3 of the text as the simplest example of how a datapath can be designed to implement an instruction set. Multi-cycle datapaths are considerably more complicated but they used in all modern computers. Discuss the advantages of multi-cycle datapaths over single cycle datapaths from both the hardware and performance perspectives.

Problem 1 - Solution

- ❑ The multi-cycle datapath is more advantageous from a hardware perspective because it requires fewer major hardware components. For example:
- ❑ A single memory for instructions and data can be used
- ❑ The same ALU can be used to calculate instruction addresses and perform arithmetic and logic instructions
- ❑ Although some additional registers and muxes are required, these are cheaper (and require less chip "real estate") than the additional memory and adders required in the single-cycle datapath.
- ❑
- ❑ Since every instruction must be completed in one clock cycle in the single-cycle datapath, this means that the clock rate depends on the execution time for the slowest instruction. (e.g. a group can only walk as fast as its slowest member if it wants to stay together) In a multiple-cycle datapath, different instructions can take a different number of short clock cycles to complete. This means that the simple instructions can be completed in a shorter time and thus the overall performance will be better, particularly since the simpler instructions are normally the most common (make the common case fast).

Problem 2

- ❑ Compare and contrast hardwired versus microprogrammed control. Discuss the advantages and disadvantages of each approach. Which approach is better for simple instruction set architectures? Which is better for complex instruction set architectures?



Problem 2 - Solution

- Microprogrammed control uses microinstructions stored in a ROM (the Control Store) to generate the datapath and memory control signals required to execute an instruction. This is in contrast to hardwired control where discrete logic gates (e.g. AND, OR and NOT gates), multiplexors and other such circuit devices are used to generate the control signals. These circuit devices are faster than a ROM and thus hardwired control is faster than microprogrammed control. For simple instruction sets, it is possible to develop an FSM for the control and implement it in discrete logic. Thus, hardwired control is more appropriate for RISC architectures. However, as the size and complexity of an instruction set increases, developing an FSM for the instruction set and then implementing it with discrete logic becomes more and more difficult. Microprogrammed control scales much better because the development of microinstruction sequences for additional, more complex instructions is a relatively straightforward process. Thus, microprogrammed control is better for CISC architectures. Another advantage of microprogrammed control is that the instruction set can be modified by changing the microprogram in the control ROM. With hardwired control, a change in the instruction set requires the redesign and replacement of the actual hardware.

Problem 3

- Explain why branch type instructions can cause problems in instruction pipelines. Also discuss how the delayed branch technique used by RISC processors like the MIPS avoids these problems with branch instructions.

Problem 3 - Solution

- ❑ Branch instructions can disrupt the flow of instructions in the pipeline. If the branch is taken, then the instructions after the branch instruction that are already in the pipeline must be purged because they shouldn't be executed.
- ❑
- ❑ The delayed branch technique avoids this problem by mandating that the instruction immediately following a branch instruction will be executed regardless of whether the branch is taken or not. This will prevent the problem discussed above as long as the pipeline is set up so that the decision on whether the branch is taken or not is made prior to the second instruction after the branch instruction being fetched.

Problem 4

- The control organization of a computer determines how the signals that initiate and control the datapath and memory operations are generated. What type of control organization is generally found in RISC processors versus that found in CISC processors? Explain why it is better to use the one type of control in RISC and the other type in CISC.

Problem 4 - Solution

- ❑ Hardwired control is normally found in RISC processors because it is faster than microprogrammed control and suitable for simple instruction sets. Microprogrammed control is normally found in CISC processors because it would be very difficult to design a hardwired control to decode and execute the numerous and complicated instructions in the CISC instruction set.

Problem 5

- Briefly discuss the hardware and software solutions to the problems caused by data hazards.

Problem 6

- In the example of page 425, we saw that the performance advantage of the multicycle design was limited by the longer time required to access memory versus use of the ALA. Suppose the memory access became 2 clock cycles long. Find the relative performance of the single-cycle and multicycle design.

Problem 7

- ❑ Describe the effect of a stuck at 0 fault would have for the signals shown below, in a multiple-data path.
- ❑ RegWrite = 0
- ❑ MemRead = 0
- ❑ MemWrite = 0
- ❑ IRWrite = 0
- ❑ PCWrite = 0
- ❑ PCWriteCond = 0

Problem 7 - Solution

- ❑ $\text{RegWrite} = 0$: All R-format instructions, in addition to lw, will not work because these instructions will not be able to write their results to the register File.
- ❑ $\text{MemRead} = 0$: None of the instructions will run correctly because instructions will not be fetched from memory.
- ❑ $\text{MemWrite} = 0$: sw will not work correctly because it will not be able to write to the data memory.
- ❑ $\text{IRWrite} = 0$: None of the instructions will run correctly because instructions fetched from memory are not properly stored in the IR register.
- ❑ $\text{PCWrite} = 0$: Jump instructions will not work correctly because their target address will not be stored in the PC.
- ❑ $\text{PCWriteCond} = 0$: Taken branches will not execute correctly because their target address will not be written into the PC.

Problem 8

- ❑ We have a program consisting of three conditional branches. The program will be executed thousands of times. Below are the outcomes of each branch for one execution of the program (T for taken, N for not taken).
- ❑ Branch 1: TNNNTN
- ❑ Branch 2: NTNNTN
- ❑ Branch 3: TNTT TN
- ❑
- ❑ Assume the behavior of each branch remains the same for each execution. For dynamic schemes assume each branch has its own prediction buffer and each buffer initialized to the same state before each execution. List the predictions for the following branch predictions schemes:
 - ❑ a) Always taken
 - ❑ b) Always not taken
 - ❑ c) 1-bit predictor, initialized to predict taken
 - ❑ d) 2-bit predictor, initialized to predict taken

Problem 8 - Solution

- ❑ Branch 1: prediction: T-T-T, right: 3, wrong: 0
- ❑ Branch 2: prediction: T-T-T-T, right: 0, wrong: 4
- ❑ Branch 3: prediction: T-T-T-T-T-T, right: 3, wrong: 3
- ❑ Branch 4: prediction: T-T-T-T-T, right: 4, wrong: 1
- ❑ Branch 5: prediction: T-T-T-T-T-T-T, right: 5, wrong: 2
- ❑ Total: right: 15, wrong: 10
- ❑ Accuracy = $100\% * 15/25 = 60\%$

Problem 9

- Using a drawing show the forwarding paths needed to execute the following instructions in a pipeline architecture:

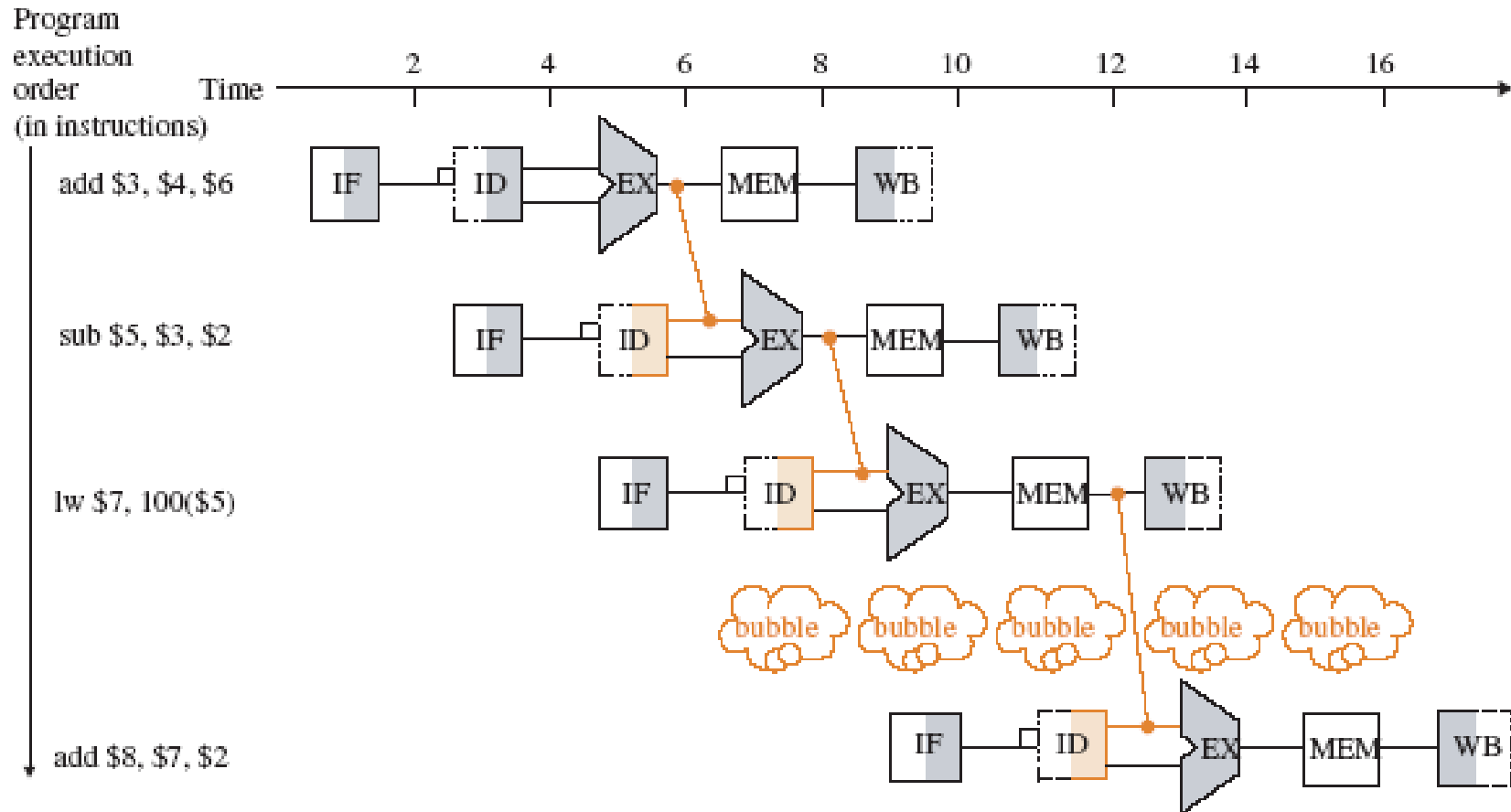
add \$3, \$4, \$16

sub \$5, \$3, \$ 2

lw \$7, 100(\$5)

add \$8, \$7, \$2

Problem 9 - Solution



Problem 10

- How many cycles will it take to execute this code in a pipelined datapath?

```
lw $4, 100($2)
```

```
sub $6, $4, $3
```

```
add $2, $3, $5
```

Problem 10 - Solution

