



[Courseware](#) [Course Info](#) [Discussion](#) [Wiki](#) [Progress](#) [Discussion Guidelines](#) [Resources](#) [Exploring Engineering](#)
[Syllabus](#) [How to Use Jade](#)

Help**PLEASE READ BEFORE DOING ILP HOMEWORK**

As is typical for our homework problems, we are asking you to apply what you have learned to conditions you may not have encountered.

After completing the problem please be sure to pay attention to the explanations in "Show Answer", where Professor Albonesi has added some further content to extend your learning.

1. HOMEWORK

Consider the following end of an LC-3 program:

```
ADD  R1, R2, R3
AND  R4, R1, R5
NOT  R5, R6
ADD  R7, R4, R5
STR  R7, R2, #0
HALT
```

Assume that these instructions are stored in an issue queue in a superscalar pipeline with dual issue capability, and that the values for R2 and R3 are stored with the first ADD, R5 with the AND, R6 with the NOT, and R2 with the STR.

Note that the data dependence between the first ADD and the AND involving R1 prevents the two instructions from issuing at the same time. While the NOT is ready to issue ahead of the AND, note the situation regarding R5. Because the NOT follows the AND, the programmer did not intend for the AND to use the value produced by the NOT. Register R5 is simply being reused for the NOT instruction. This is called a *Write After Read (WAR)* hazard.

1 A. HOMEWORK (1/1 point)

Which of the following describes how the hardware can handle this case in order to ensure correct program execution?

- ☐ Allow the NOT to issue ahead of the AND and bypass its result to the AND.
- ☐ Allow the first ADD and the AND to issue at the same time.
- ☒ Prevent the NOT from issuing ahead of the AND. ✓
- ☐ Prevent the AND from issuing ahead of, or at the same time as, the NOT.

EXPLANATION

The first option is what we want to avoid; the NOT should not bypass its result to the ADD.

The second option cannot work due to the data dependence involving R1.

The third operation will ensure correct operation since it will prevent the bypassed R5 from being used by the AND. Note that the AND and the NOT can issue together.

The fourth option is actually what we want to happen in order for the code to work, so we shouldn't prevent it from occurring.

Final Check

Save

Hide Answer


You have used 1 of 2 submissions

Show Discussion

 New Post

1 B. HOMEWORK (1/1 point)

How can the assembly language programmer change this code to allow the hardware to issue the NOT to ahead of the AND while ensuring correct operation?

- ☐ Switch the order of the first ADD and the AND.
- ☐ Switch the order of the AND and the NOT.
- ☐ Change the second source register of the AND from R5 to something different, e.g., R7.
- ☒ Change the destination register of the NOT, and the second source register of the second ADD, to R7. 

EXPLANATION

The first option cannot work due to the data dependence involving R1.

The second option will cause the NOT to bypass its result to the ADD, which is what we want to avoid.

R5 would have been produced previously in the program, so changing it to R7 will change the operation of the program.

The fourth option eliminates the WAR hazard, which allows the hardware to issue the NOT ahead of the AND. However, we need to change the second AND as well to preserve the operation of the original program.

Final Check

Save

Hide Answer

You have used 1 of 2 submissions

Show Discussion

 New Post

Help

1 C. HOMEWORK (1/1 point)


A hardware technique called register renaming is used within smartphone processors to handle WAR hazards without the programmer needing to make changes to the code. The idea is to implement more physical registers than are specified in the ISA and to add a pipeline stage between ID and the issue queue in which the logical addresses in the program (those defined by the ISA and used by the compiler) are converted by the hardware into physical addresses. In this process, data dependencies are preserved and each instruction is assigned a unique physical destination register, which eliminates WAR hazards.

For example, an LC-3 design using register renaming might implement 16 physical registers and rename the original instruction sequence as follows:

```
ADD  p1, p2, p3
AND  p4, p1, p5
NOT  p7, p6
ADD  p8, p4, p7
STR  p8, p2, #0
HALT
```

Here the logical registers Rx are converted by the hardware to physical register names py (x may be different than y, where x and y are numbers). Notice that data dependencies between instructions are preserved, but WAR hazards are avoided by assigning each instruction a different physical destination register. This eliminates the WAR hazard between the AND and the NOT.

How would the issue queue with dual issue capability handle the first four instructions of this sequence with register renaming?

- ☐ The four instructions would issue one by one in four cycles.
- ☐ The first ADD and the AND would issue together, followed by the NOT and second ADD in the second cycle.
- ☐ The first ADD and the NOT would issue together, followed by the AND and second ADD in the second cycle.
- ☒ The first ADD and the NOT would issue together, followed by the AND in the second cycle, and the second ADD in the third cycle. 

EXPLANATION

Due to the data dependence involving p1 between the first ADD and the AND, they cannot issue together. However, the NOT does not depend on either of these instructions, and register renaming has eliminated the WAR hazard with the AND. Therefore, the NOT can issue together with the first ADD. The NOT and the second ADD cannot issue together due to the data dependence involving p7. Therefore, they need to issue on back to back cycles.

[Final Check](#)[Save](#)[Hide Answer](#)

You have used 1 of 2 submissions

[Show Discussion](#)[New Post](#)



Help



EdX offers interactive online classes and MOOCs from the world's best universities. Online courses from MITx, HarvardX, BerkeleyX, UTx and many other universities. Topics include biology, business, chemistry, computer science, economics, finance, electronics, engineering, food and nutrition, history, humanities, law, literature, math, medicine, music, philosophy, physics, science, statistics and more. EdX is a non-profit online initiative created by founding partners Harvard and MIT.

© 2015 edX Inc.

EdX, Open edX, and the edX and Open edX logos are registered trademarks or trademarks of edX Inc.

[Terms of Service and Honor Code](#)

[Privacy Policy \(Revised 10/22/2014\)](#)



About edX

[About](#)

[News](#)

[Contact](#)

[FAQ](#)

[edX Blog](#)


[Donate to edX](#)

[Jobs at edX](#)


Follow Us


 [Facebook](#)


 [Twitter](#)


 [LinkedIn](#)

 [Google+](#)

 [Tumblr](#)

 [Meetup](#)

 [Reddit](#)

 [Youtube](#)