MITx 6.004.3x

Computation Structures 3: Computer Organization

Help selfpoised

Course Progress Dates Discussion

Course / 15. Pipelining the Beta / Lecture Videos (50:06)

‹ Previous Next ›

LE15.2

Bookmark this page

Calculator

For all Beta related questions, you should make use of the Beta documentation, the Beta Instruction Summary, the Unpipelined Beta Diagram and the Pipelined Beta Diagram.

LE15.2.1: Pipelined Execution

0.0/1.0 point (ungraded)

You are exploring several different Beta implementations, using a benchmark program that includes the simple loop shown below. Note that the code contains a number of **NOP()** instructions, defined as **ADD(R31,R31,R31)**.

```
        ...
        NOP() NOP() NOP() NOP()
Loop:
        LD(R0, 0, R0)
AA:
        MUL(R0, R1, R4)
BB:
        BNE(R0, LOOP)
CC:
        ADD(R0, R3, R3)
        NOP() NOP() NOP() NOP()
        ...
```

Your first experiment is to run the code on an unpipelined Beta, like the one shown in Lecture (and studied in the lab assignment).

You observe that the program runs through several iterations of the loop, before dropping out and executing the **ADD** and subsequent instructions.

1. On an **unpipelined** Beta, how many clock cycles of execution time are required **for each iteration** through the loop?

   Unpipelined Beta, clocks per loop iteration:

   Next, you run the code on a **fully functional 5-stage Beta pipeline** (with working bypass, annul, and stall logic) and count the cycles per loop iteration. Note that the code scrap begins and ends with sequences of **NOPs**; thus you don't need to worry about pipeline hazards involving interactions with instructions outside of the region shown.

2. How many clock cycles of execution time are required by the fully functional 5-stage pipelined Beta **for each iteration** through the loop?

   Functional 5-stage pipelined Beta, clocks per loop iteration:

   Finally, you try to run the code on a **defective** version of the 5-stage pipelined Beta: **it has no bypass paths, annulment of instructions in branch delay slots, or pipeline stalls**. You undertake to convert the above code to run on the defective processor, by adding the **minimum** number of **NOP** instructions at the various tagged points in this code to make it give the same results on your defective pipelined Beta as it gives on a normal, unpipelined Beta.

3. Specify the minimal number of **NOP** instructions (defined as **ADD(R31,R31,R31)**) to be added at each of the labeled points in the above program to make it work properly on the defective 5-stage Beta pipeline.

NOPs at Loop:

NOPs at AA:

NOPs at BB:

Calculator

NOPs at CC:

Submit

LE15.2.2: Branch delay slots

0.0/1.0 point (ungraded)

This is a very tricky, classic pipelining problem.

The following instruction sequence is executed on a 5-stage pipelined Beta with bypass paths **but no branch delay slot annulment**, i.e., the instruction following a branch is always executed whether the branch is taken or not.

```
      CMOVE(5, R0)          | Load 5 into R0
      CMOVE(7, R1)          | Load 7 into R1

      BR(X)
      BR(Y)

Y:    ADDC(R0, 1, R0)       | Increment R0 contents by 1

X:    SUBC(R1, 1, R1)       | Decrement R1 contents by 1

      NOP()                 | wait, to stabilize registers
      NOP()
      NOP()
      HALT()
```

Your challenge is to figure out the final values in R0 and R1.

Final R0 Value:

Final R1 Value:

Submit

Discussion

Hide Discussion

Topic: 15. Pipelining the Beta / LE15.2

Add a Post

Show all posts ⌄ by recent activity ⌄

Is BNE an instruction?
Hi, is BNE an instruction, because the answer to unpipelined beta, seems to rule out BNE? 2

LE 15.2.1 Defective Pipeline Answer
Why is the MUL instruction able to read R0 in cycle 5? I thought that with no functional bypass logic, MUL would need to wait until cy... 8

✓ LE15.2.1: Pipelined Execution: How does it drop out of the loop? 5

< Previous Next >

Calculator

edX

[About](#)
[Affiliates](#)
[edX for Business](#)
[Open edX](#)
[Careers](#)
[News](#)

Legal

[Terms of Service & Honor Code](#)
[Privacy Policy](#)
[Accessibility Policy](#)
[Trademark Policy](#)
[Sitemap](#)

Connect

[Blog](#)
[Contact Us](#)
[Help Center](#)
[Media Kit](#)
[Donate](#)

GET IT ON Google play

Download on the App Store

Calculator