
**Course Name**: Computer Architecture Lab

**Course Number and Section**: 14:332:333:03

**Experiment**: [Experiment # [6] – CPU Structure, Pipeline Programming and Hazards]

**Lab Instructor**: Christos Mitropoulos

**Date Performed**: November 28 2018

**Date Submitted**: December 12 2018

**Submitted by**: Vancha Verma 173004061

**Course Name**: Computer Architecture Lab

**Course Number and Section**: 14:332:333:03

**! Important: Please include this page in your report if the submission is a paper submission. For electronic submission (email or Sakai) please omit this page.**

----------For Lab Instructor Use ONLY----------

GRADE: _____

COMMENTS:

**ASSIGNMENT 1**

**Suppose you have the CPU of Figure 1 and that the architecture is not pipelined. For the following program indicate the values of the control signals for each instruction.**

```
addi t0, t0, 10
lb t1, 32(t0)
ori t2, t0, 4
bne t2, t0, exit
```

| Instruction | ALUSrc | MemtoReg | RegWrite | MemRead | MemWrite | Branch | ALUOp |
|---|---|---|---|---|---|---|---|
| `addi t0, t0, 10` | 1 | 0 | 1 | 0 | 0 | 0 | 10 |
| `lb t1, 32(t0)` | 1 | 1 | 1 | 1 | 0 | 0 | 00 |
| `ori t2, t0, 4` | 1 | 0 | 1 | 0 | 0 | 0 | 10 |
| `bne t2,t0, exit` | 0 | 0 | 0 | 0 | 0 | 1 | 01 |

**ASSIGNMENT 2**

**Suppose you have a machine that supports the five pipeline stages mentioned above. Using the following program to answer the questions:**

```
lw t0, 0(t3)
add t0, t0, t1
sub t1, t3, t1
addi t4, t3, 4
sub t5, t5, t4
```

**1. Complete the table of execution with five pipeline stages. Ignore all hazards**

| Instruction/ Cycle | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|
| `lw t0,0(t3)` | IF | ID | EX | MEM | WB | | | | |
| `add t0,t0,t1` | | IF | ID | EX | MEM | WB | | | |
| `sub t1,t3,t1` | | | IF | ID | EX | MEM | WB | | |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| `addi t4,t3,4` | | | | IF | ID | EX | MEM | WB | |
| `sub t5,t5,t4` | | | | | IF | ID | EX | MEM | WB |

**2. Suppose that register t0, t1, t2, t3, t4, and t5 have the respective values of 2, 5, 8, 2, 4, 1. Give the values of the registers after running the program without the processor handling any hazards.**

t0 = 7, t1= -3, t2 = 8, t3 = 2, t4 = 6, t5 = -3

**3. Identify the hazards and explain their nature;**

Since load word is not done writing in t0, it will cause a hazard in add, which called on t0. There will be another hazard after addi. When sub calls on t4, addi won't be done writing it back.

**4. Insert the necessary NOP instructions so that the registers would get the values when the processor executes the program one instruction per cycle.**

| **Instruction/ Cycle** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **11** | **12** | **13** |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| `lw t0,0(t3)` | IF | ID | EX | MEM | WB | | | | | | | | |
| `NOP` | | | | | | | | | | | | | |
| `NOP` | | | | | | | | | | | | | |
| `add t0,t0,t1` | | | | IF | ID | EX | MEM | WB | | | | | |
| `sub t1,t3,t1` | | | | | IF | ID | EX | MEM | WB | | | | |
| `addi t4,t3,4` | | | | | | IF | ID | EX | MEM | WB | | | |
| `NOP` | | | | | | | | | | | | | |

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| `NOP` | | | | | | | | | | | | | |
| `sub t5,t5,t4` | | | | | | | | | IF | ID | EX | MEM | WB |

**5. Discuss how instruction reordering can avoid the hazards in the code section above.**

By rearranging the instructions, we are able to reduce the number of stalls. For example, t0 from lw isn't called till after addi. Rearranged code:

```
lw t0, 0(t3)
addi t4, t3, 4
add t0, t0, t1
sub t1, t3, t1
sub t5, t5, t4
```

**6. Discuss how forwarding can affect the execution of the code section above. Compare it with NOP and instruction reordering in terms of number of cycles needed to complete**.

Forwarding reduces stalls due to data hazard to 1 stall rather than 2 stalls. As there are two data hazards, there will only be 2 more cycles. There are 4 extra cycles when using NOP. Reordering instructions will reduce the cycles furthermore as it avoids hazards. Thus, with forwarding there will only 10 cycles.

**ASSIGNMENT 3**

**Consider the following code:**

```
addi t1, t0, 3
Loop: addi t2, t1, 1
sub t0, t0, t1
bne t0, zero, loop
```

**Assume that t0 has initial value of -6**

**1. Determine how many cycles would it take for the program to execute without hazards handling hardware.**

11 cycles to execute without hazards.

**2. Draw the execution table and determine the type of hazards**

| **Instruction/Cycle** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **11** |
|---|---|---|---|---|---|---|---|---|---|---|---|
| `addi t1,t0,3` | IF | ID | EX | MEM | WB | | | | | | |
| `Loop: addi t2,t1,1` | | IF | ID | EX | MEM | WB | | | | | |
| `sub t0,t0,t1` | | | IF | ID | EX | MEM | WB | | | | |
| `bne t0,zero, loop` | | | | IF | ID | EX | MEM | WB | | | |
| `Loop: addi t2,t1,1` | | | | | IF | ID | EX | MEM | WB | | |
| `sub t0,t0,t1` | | | | | | IF | ID | EX | MEM | WB | |
| `bne t0,zero, loop` | | | | | | | IF | ID | EX | MEM | WB |

The data hazards are written in red font in the table above. In all cases, the previous instructruction hasn't been completed yet and thus, the incorrect value is being used from the register.

**3. Write program using NOP instructions to avoid the hazards**

```
addi t1, t0, 3
NOP
NOP
Loop: addi t2, t1, 1
sub t0, t0, t1
NOP
NOP
bne t0, zero, loop
```

**4. Describes the effects of instruction forwarding to this code.**

Forwarding would allow for the code to execute with only one stall for every data hazard. On the other hand, without forwarding there will be 2 stalls for every Data Hazard. Therefore, forwarding instructions reduces the number of cycles used to execute the code.

Total cycles with forward: 11 + (1*3) = 14 cycles
Total cycles without forward: 11 + (2*3) = 17 cycles

**ASSIGNMENT 4**

**File "quad_sol.s" contains a quadratic polynomial solver, which calculates the integer solution of a quadratic polynomial equation.**

**1. Rewrite the program using instructions reordering to reduce the number of cycles needed to execute the program. Indicate the number of cycle reduction.**

```
     li t3, 2                # c=2, moved
     li t1, 1           # a=1
     li t2, -3          # b=-3
# In the following lines all the necessary steps are taken to
# calculate the discriminant of the quadratic equation
# D = b^2 - 4*a*c

mul t5,t1,t3 # t5 = t1*t3, where t1 holds a and t3 holds c,
moved
mul t4,t2,t2        # t4 = t2*t2, where t2 holds b
li a3, 4            # moved
mul t5,t5,a3        # Multiply value of s0 with 4, creating
4*a*c
li s0, 1            # Square Root Partial Result, sqrt(D), moved
sub t6,t4,t5        # Calculate D = b^2-4*a*c
. . . .

     endsqrt:

     li t0, 2  # Load constant number to integer register,
movedd
     neg s2,t2 # calculate -b and save it to s2
     mul s5,t1,t0   # Calculate 2*a and save it to s5, moved
```

The moved instructions are marked with a 'moved' in the comments. The previous code had 2 stalls for every hazard. After reordering the code, I have been able to reduce it to 1 stall for every hazard. Therefore, the number of cycles added due to stalls have now been divided by 2.

**2. Describe how forwarding would affect the execution of the program.**

With the addition of forward, there will only be one stall required to execute the program. Thus, reducing the total number of cycles even more.