CS1216 - Monsoon 2022 - Homework 5

Jivansh Sharma
UG 24 1020211193

21/11/2022

Collaborators: None

1. Consider an in-order 5-stage pipeline similar to the one discussed in class. First assume that the pipeline does not support bypassing (forwarding). What are the stall cycles introduced between the following pairs of back-to-back instructions? Then, solve the same problem while assuming support for bypassing. Show your work clearly i.e., show how each instruction goes through the 5 stages, indicate the point of production and point of consumption, show where consuming instruction is held back when there are stalls. Recall that a register read is performed in the second half of the D/R stage and a register write is performed in the first half of the RW stage

```
A)  lw $5, 64($7)
    add $4, $5, $3
```

```
Without Forwarding
```
Cycles: 1 2 3 4 5 6 7

```
lw would go in the following stages:
IF D/R ALU DM  RW

and add would go in the following stages:
IF STALL STALL D/R ALU DM RW
```

The add instruction consumes 2 stall cycles in D/R stage as it waits for lw to write to $5. lw finishes writing in first half of cycle 5 and add picks it up from the second half.

```
With Forwarding
```
Cycles: 1 2 3 4 5 6 7

```
lw would go in the following stages:
IF D/R ALU DM  RW
```

```
and add would go in the following stages:
IF STALL D/R ALU DM RW
```

In this case, at the 4th cycle the address is at the ALU stage and loads it into the DM stage for add. Then in the 5th clock cycle, the operand is available.

```
B)  lw $1, 32($2)
    sw $1, 128($5)
```

```
Without Forwarding
```
Cycles: 1 2 3 4 5 6 7 8

```
lw would go in the following stages:
IF D/R ALU DM  RW

and sw would go in the following stages:
IF STALL STALL D/R ALU DM RW
```

The lw instruction consumes 2 stall cycles in D/R stage as it waits for lw to write to $1. lw finishes writing in first half of cycle 1 and add picks it up from the second half.

```
With Forwarding
```
Cycles: 1 2 3 4 5 6 7

```
lw would go in the following stages:
IF D/R ALU DM  RW

and sw would go in the following stages:
IF STALL D/R ALU DM RW
```

In this case, at the 4th cycle the address is at the ALU stage and loads it into the DM stage for sw.

```
B)  lw $1, 32($2)
    sw $1, 128($5)
```

```
Without Forwarding
```
Cycles: 1 2 3 4 5 6 7 8

```
lw would go in the following stages:
IF D/R ALU DM  RW

and sw would go in the following stages:
IF STALL STALL D/R ALU DM RW
```

The lw instruction consumes 2 stall cycles in D/R stage as it waits for lw to write to $1. lw finishes writing in first half of cycle 5 and add picks it up from the second half.

```
With Forwarding
```
Cycles: 1 2 3 4 5 6 7

```
lw would go in the following stages:
IF D/R ALU DM  RW

and sw would go in the following stages:
IF STALL D/R ALU DM RW
```

In this case, at the 4th cycle the address is at the ALU stage and loads it into the DM stage for sw.

```
C)  add $1, $2, $3
    sub $4, $2, $1

Without Forwarding
```
Cycles: 1 2 3 4 5 6 7 8

```
add would go in the following stages:
IF D/R ALU DM  RW

and sub would go in the following stages:
IF STALL STALL D/R ALU DM RW
```

The sub instruction consumes 2 stall cycles in D/R stage as it waits for add to write to $1. add finishes writing in first half of cycle 5 and sub picks it up from the second half.

```
With Forwarding
```
Cycles: 1 2 3 4 5 6

```
add would go in the following stages:
IF D/R ALU DM  RW

and sub would go in the following stages:
IF D/R ALU DM RW
```

There is no stalling now, the shared value is computed in ALU stage of add instruction and passed to sub in cycle 4.

2. Consider an in-order pipeline that has the same stages as a 5 stage pipeline discussed in class. However, unlike the 5-stage pipeline discussed in class, a register read in this design takes an entire cycle and a register write takes an entire cycle (and not a half cycle).
Use the same sequence of instructions from Question 1, and show how these sequences proceed through the pipeline. Indicate the number of stall cycles that are experienced by each sequence. Show your work for both versions of the pipeline: one without bypassing, and another one with bypassing.

```
A)  lw $5, 64($7)
    add $4, $5, $3
```

```
Without Forwarding
```
Cycles: 1 2 3 4 5 6 7 8 9

```
lw would go in the following stages:
IF D/R ALU DM  RW

and add would go in the following stages:
IF STALL STALL STALL D/R ALU DM RW
```

The add instruction consumes 3 stall cycles in D/R stage as it waits for lw to write to $5. This now takes an additional cycle as compared to q1 hence the increase to 3 cycles.

```
With Forwarding
```
Cycles: 1 2 3 4 5 6 7

```
lw would go in the following stages:
IF D/R ALU DM  RW

and add would go in the following stages:
IF STALL D/R ALU DM RW
```

In this case, at the 4th cycle the address is at the ALU stage and loads it into the DM stage for add. Then in the 5th clock cycle, the operand is available.

```
B)  lw $1, 32($2)
    sw $1, 128($5)
```

```
Without Forwarding
```
Cycles: 1 2 3 4 5 6 7 8 9

```
lw would go in the following stages:
IF D/R ALU DM  RW

and sw would go in the following stages:
IF STALL STALL STALL D/R ALU DM RW
```

The lw instruction consumes 3 stall cycles in D/R stage as it waits for lw to write to $1. lw finishes writing in cycle 5 and add sw picks it up from the cycle 6.

```
With Forwarding
```
Cycles: 1 2 3 4 5 6 7

```
lw would go in the following stages:
IF D/R ALU DM  RW

and sw would go in the following stages:
IF STALL D/R ALU DM RW
```

In this case, at the 4th cycle the address is at the ALU stage and loads it into the DM stage for sw in the 5th cycle.

```
C)  add $1, $2, $3
    sub $4, $2, $1

Without Forwarding
```
Cycles: 1 2 3 4 5 6 7 8 9

```
add would go in the following stages:
IF D/R ALU DM  RW

and sub would go in the following stages:
IF STALL STALL STALL D/R ALU DM RW
```

The sub instruction consumes 3 stall cycles in D/R stage as it waits for add to write to $1. add finishes writing in cycle 5 and sub picks it up from the cycle 6.

```
With Forwarding
```
Cycles: 1 2 3 4 5 6

```
add would go in the following stages:
IF D/R ALU DM  RW

and sub would go in the following stages:
IF D/R ALU DM RW
```

There is no stalling now, the shared value is computed in ALU stage of add instruction (Cycle 3) and passed to sub in cycle 4.