Learning By Thinking

The "machine" (chapter4-1 slide 29) is great (it works for all five addressing modes)!

Q. what are those five address modes?

↓

But, Not all the hardware parts are used all the time! (what a waste!)

↓

Assembly line concept (e.g. Automobile Industry)

Q. What is the ~~at~~ main advantage of the assembly line concept?

A. "through put" is very high!

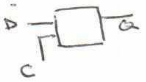
↓

Q. Can we divide the "machine of Chapter4-1 slide 29) into several stages?

A. It is not easy! Sometimes we need to have a total separation. Sometimes we have to connect two adjancent stages.

↓

Idea:



D Latch
Truth table

D	C	Q
0 1	0 0	previous Q previon Q
0	↑	0
1	↑	1

↖ Rising Edge of the control/clock Signal

↓

Great! We have a pipe line machine!

↓

Q. Are you sure?

A. We can feed the machine with instructions in a pipeline ~~fashion~~ (i.e., one after another)

Q. Are we done with this new machine?

A. Oops! The following situation creates great problems for us:

```
and $2, $3, $4
or  $6, $2, $5
add $7, $8, $2
  :
```

↓

Data Hazard

↓

Q How to solve ~~this~~ problem?

Approach:

Analyzing the problem using "Single cycle ~~block~~ Pipeline diagram

Q There are two ways of analyzing the problem. Which one is better?

A. The same problem can be analyzed by

1. single cycle pipeline diagram
2. multi-cycle Pipeline diagram

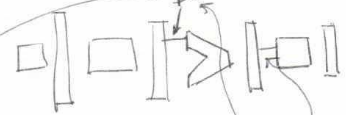
We prefer the single-cycle pipeline diagram ↓ bottom page

↓ From Top page

Observation: When the "$2" of the instruction or $6, $2, $5 reaches CC#4, ~~its~~ its value is wrong

CC #4

$2 of or $6,$2,$5



This value is wrong

↓

$1 B dollar question:

Q At CC#4, we know $2 is wrong. Do we have a correct value of $2 some wher in the system?

A. Yes!

Q. wher?

A. Here

Q. How can we bring this correct value of $2 to the place where the wrong value of $2 is in?

A. Easy! Just use a piece of wire (well, actually 32 pieces of wire) to connect two places.

Q ~~Any name~~ How to name this great invention?

A. Forwarding (Q. Is this a good name?
A. Not really. We/I prefer to call it rerouting)

↓

Q. This example shows the invention that Solve the data hazard between the first instruction and the second instruction. ~~Do~~ How do we solve the data hazard problem between the first instruction and the third instruction?

A. Another piece of wire!

↓

Q. You really can't connect wire directy to another wire. Any solution?

A. Multiplexer

↓

control of the multiplexer

↓

Top 1% Slide → Slide 22 of Chapter 4 part 3

⋮

EX/MEM.RegisterRd = ID/EX.Register.Rs

⋮

↓ To page 3

↓ From page 2

Q. So far we have "invented" two wires that solve the so-called data hazard problem as illustrated in a special example as ~~bet to~~ below:

```
and $2,$3,$4
or  $6,$2,$5
add $7, $8, $2
  :
```

→ see slide 20 of Chapter 4 part 3 → Top 1/3 slide

Q. Do those two "wires" solve <u>ALL</u> the data hazard problems?

A. NO!

Q. How come?

A. For example

```
lw $2, 32($1)
or $6, $2, $5
```

~~If~~

Observation: lw is a tough cookie!
It can't be solved ~~with~~ by ~~our "two~~ the forwarding approach.

↓

Q. In this case What do we do?

A. Hardware solution → Insert ~~a~~ one NOP
Software Solution → Insert one NOP

design extra hardware to detect "lw" case and insert one NOP (~~bla~~. blank operation) ~~bet~~ after lw.

↓

~~Hard~~
Hazard
Detection
Unit

Compiler to detect the lw problem and insert one NOP instruction after the lw instruction