


Assignment Project Exam Help

Add WeChat powcoder

L15_1 Detect-and-Forward

EECS 370 – Introduction to Computer Organization – Fall 2020



Assignment Project Exam Help

Learning Objectives

Add WeChat powcoder

- To understand the flow of data through the datapath and between pipeline stages in reducing stalls arising from data hazards.

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

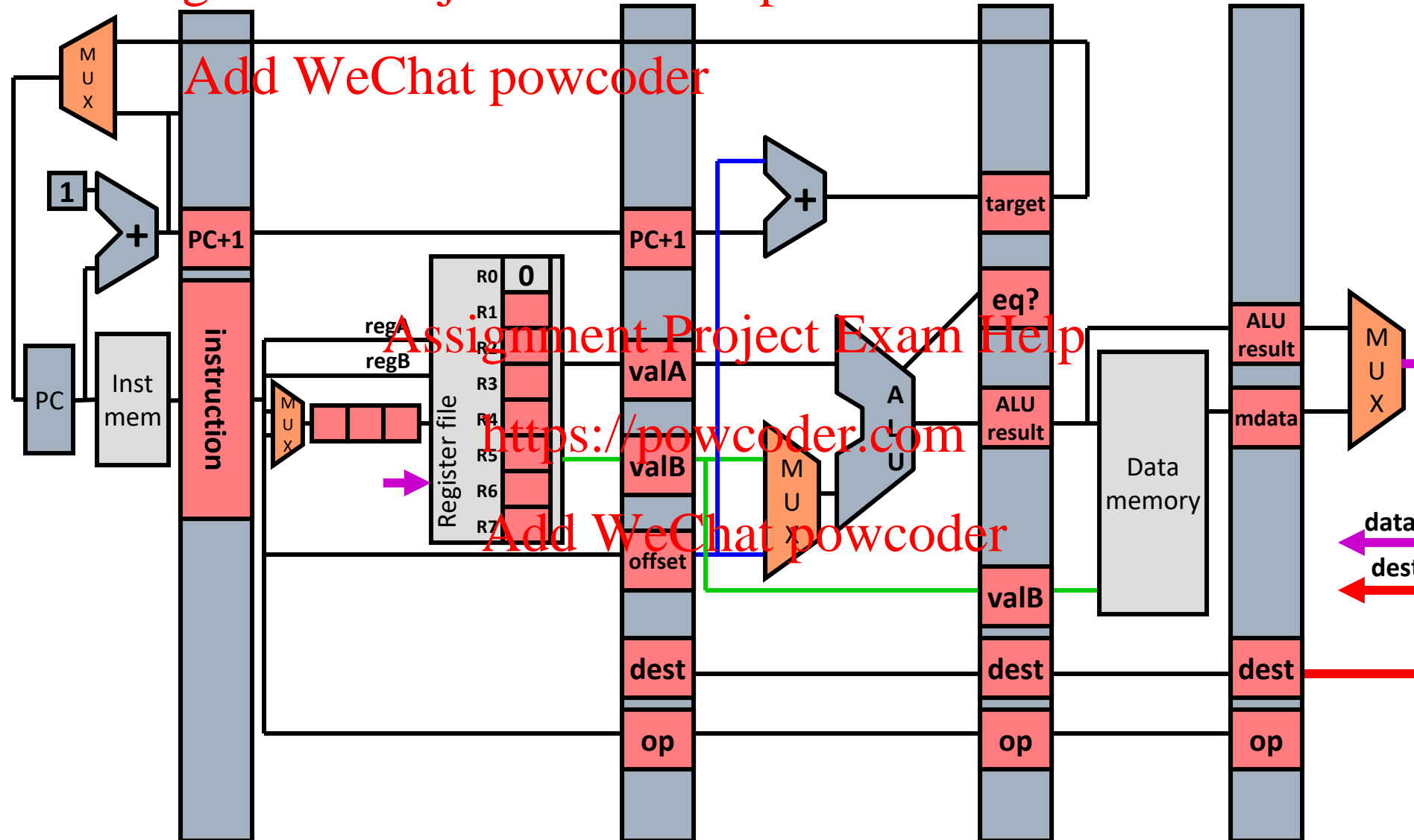
Detect and Stall - Problems



- CPI increases every time a hazard is detected!
- Is that necessary? Not always!

- Re-route the result of the add to the nor
 - nor no longer needs to read R3 from reg file
 - It can get the data later (when it is ready)
 - This lets us complete the decode this cycle
 - But we need more control to remember that the data that we are not getting from the reg file at this time will be found elsewhere in the pipeline at a later cycle.

Assignment Project Exam Help



Assignment Project Exam Help

Handling Data Hazards III: Detect and Forward

Add WeChat powcoder



- Detect: same as detect and stall
 - Except that all 4 hazards must be treated differently
 - i.e., you can't logical-OR the 4 hazard signals
- Forward:
 - New **bypass datapaths** route computed data to where it is needed
 - New MUX and control to pick the right data
- **Beware:** Stalling may still be required even in the presence of forwarding

Assignment Project Exam Help

Sample Code (Simple)

Add WeChat powcoder

Data Hazards

We will use this program for the next example

```
1. add 1 2 3
2. nor 3 4 5
3. add 6 3 7
4. lw 3 6 10
5. sw 6 2 12
```

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

HAZARDS:

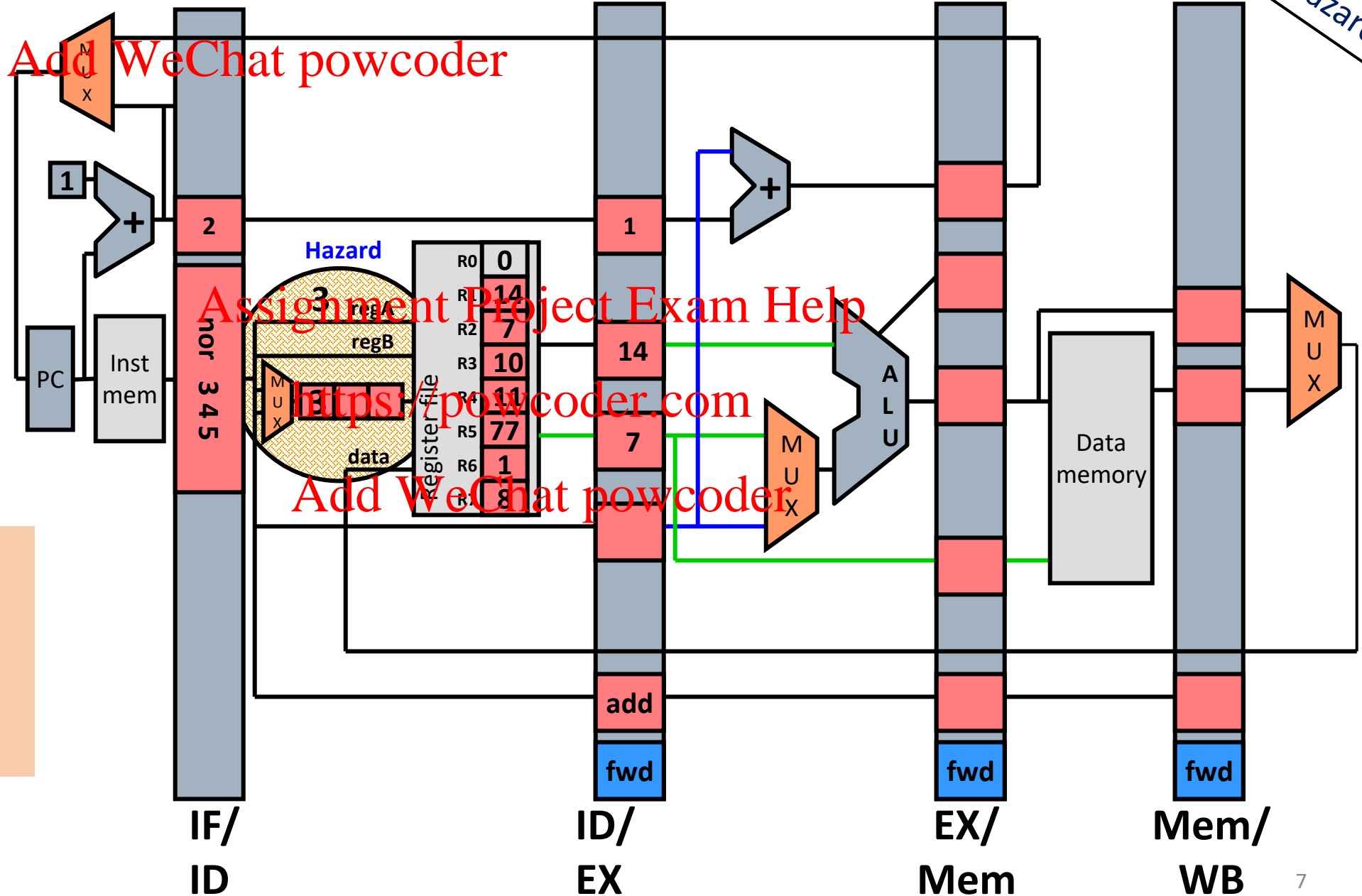
```
1. add 1 2 3
2. nor 3 4 5
3. add 6 3 7
4. lw 3 6 10
5. sw 6 2 12
```

Assignment Project Exam Help

First half of cycle 3

Add WeChat powcoder

Data Hazards

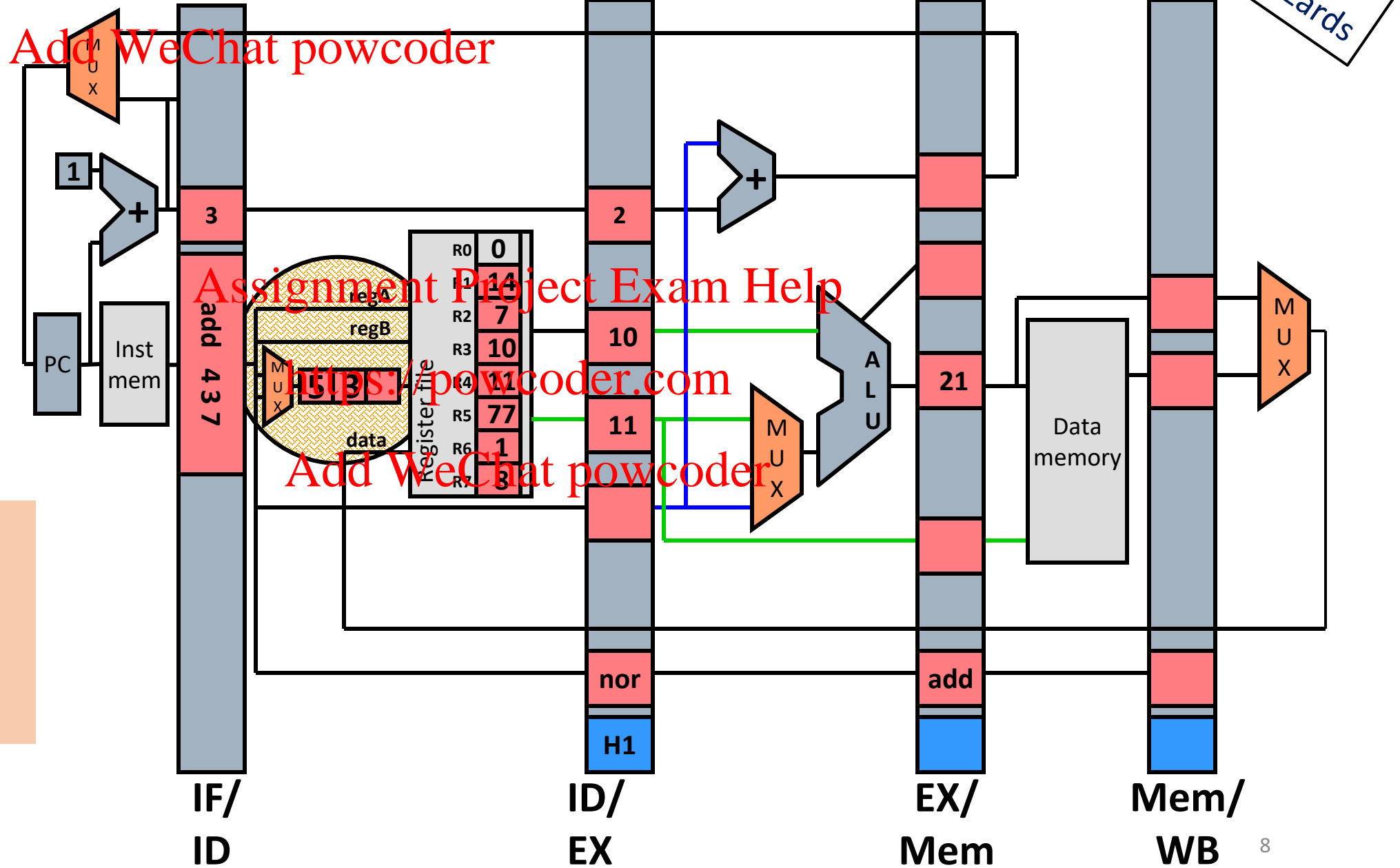


1. add 1 2 3
2. nor 3 4 5
3. add 6 3 7
4. lw 3 6 10
5. sw 6 2 12

Assignment Project Exam Help

End of cycle 3

Add WeChat powcoder



1.	add	1	2	3
2.	nor	3	4	5
3.	add	6	3	7
4.	lw	3	6	10
5.	sw	6	2	12

Assignment Project Exam Help

First half of cycle 4

Data Hazards

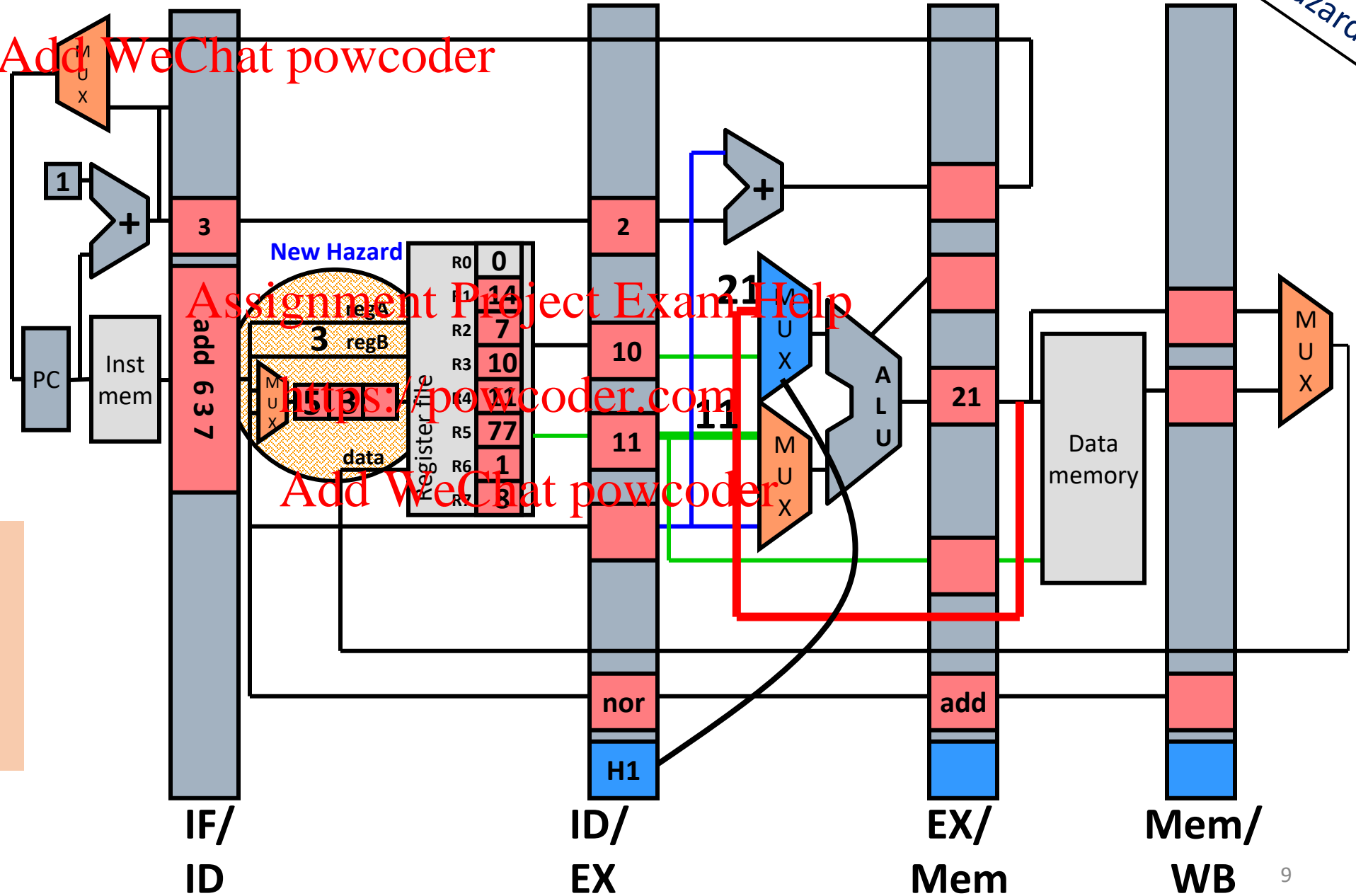
Add WeChat powcoder

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

1. add 1 2 3
2. nor 3 4 5
3. add 6 3 7
4. lw 3 6 10
5. sw 6 2 12



Assignment Project Exam Help

End of cycle 4

Data Hazards

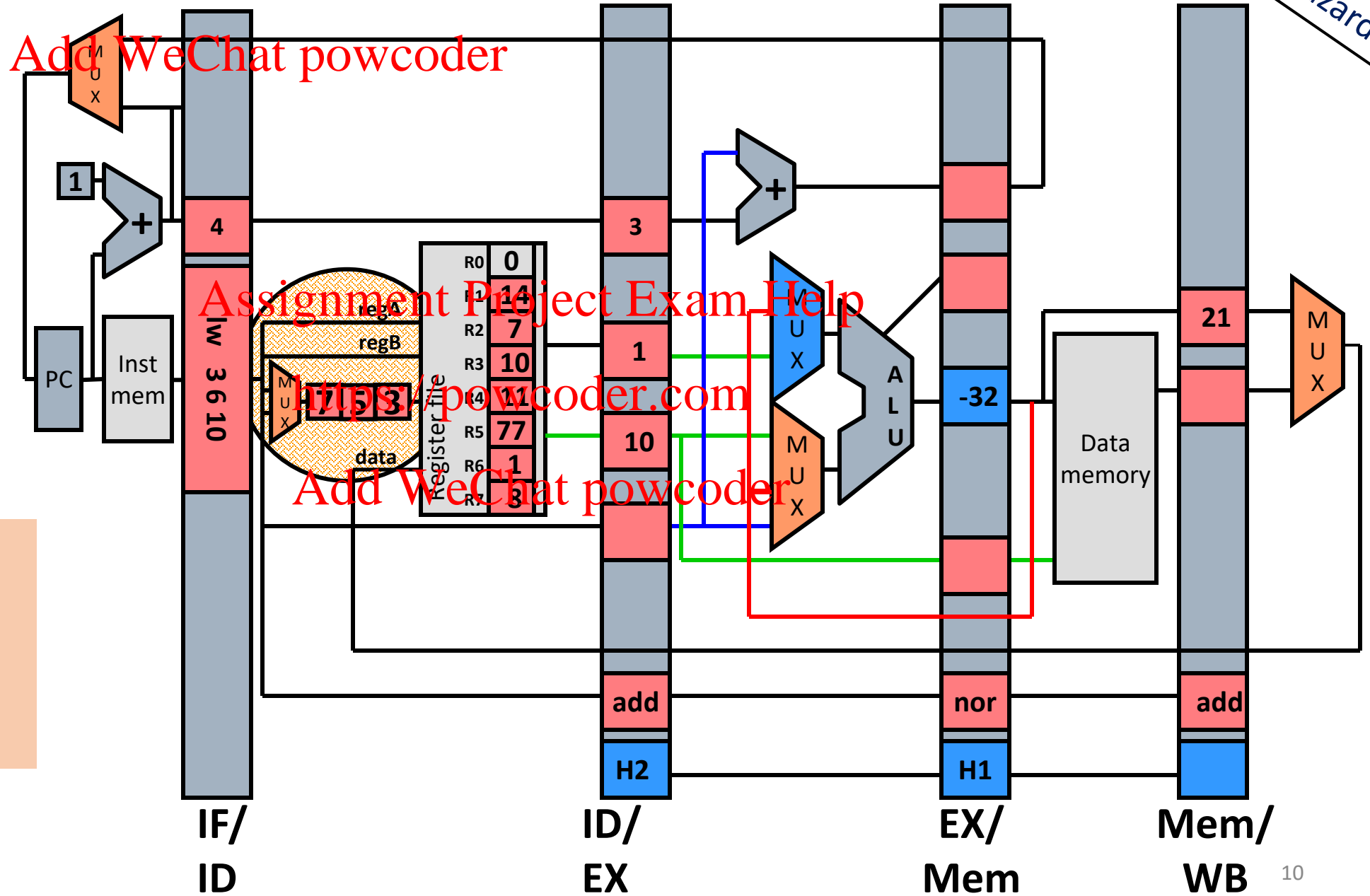
Add WeChat powcoder

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

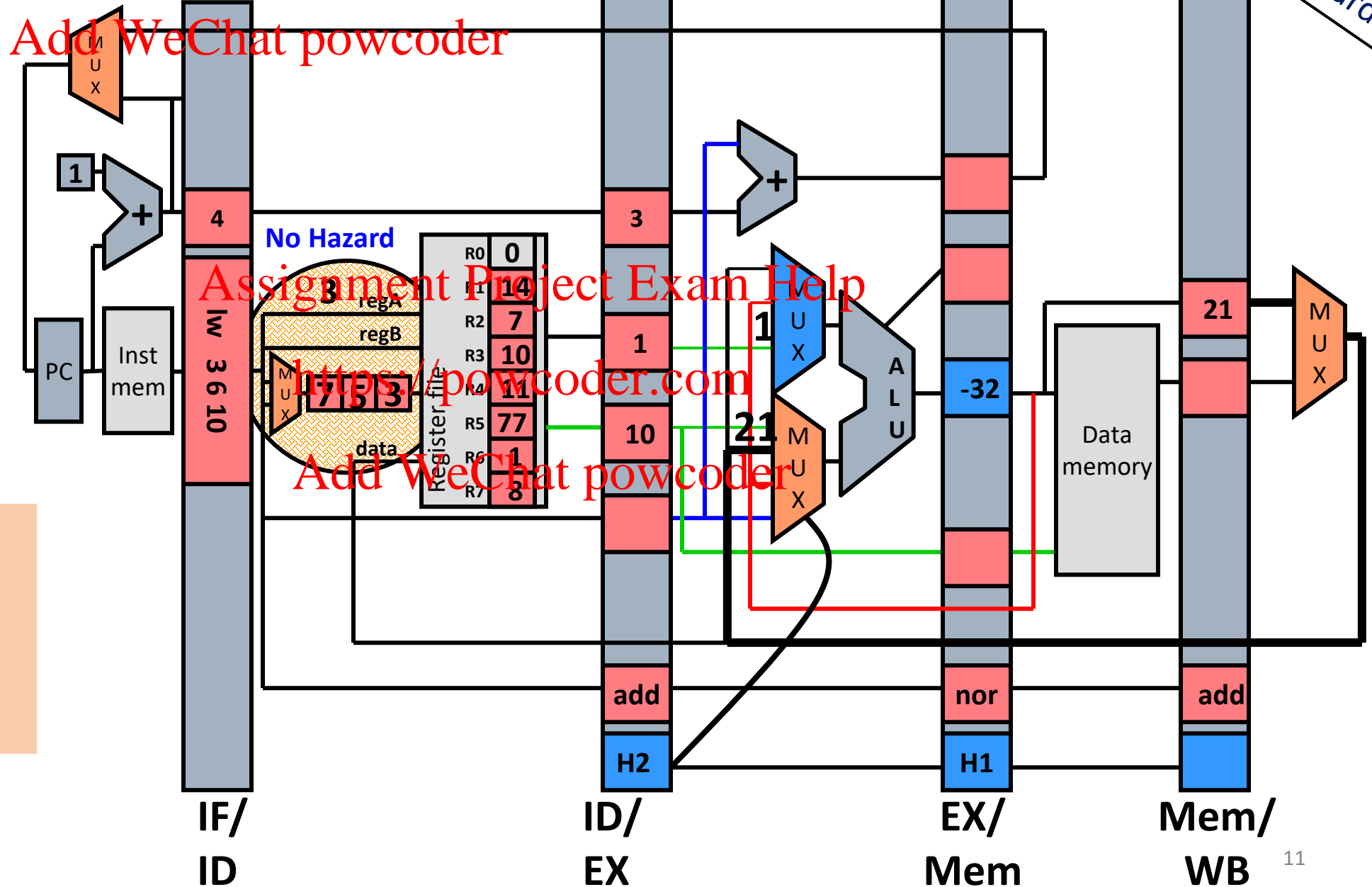
1. add 1 2 3
2. nor 3 4 5
3. add 6 3 7
4. lw 3 6 10
5. sw 6 2 12



Assignment Project Exam Help

First half of cycle 5

Data Hazards



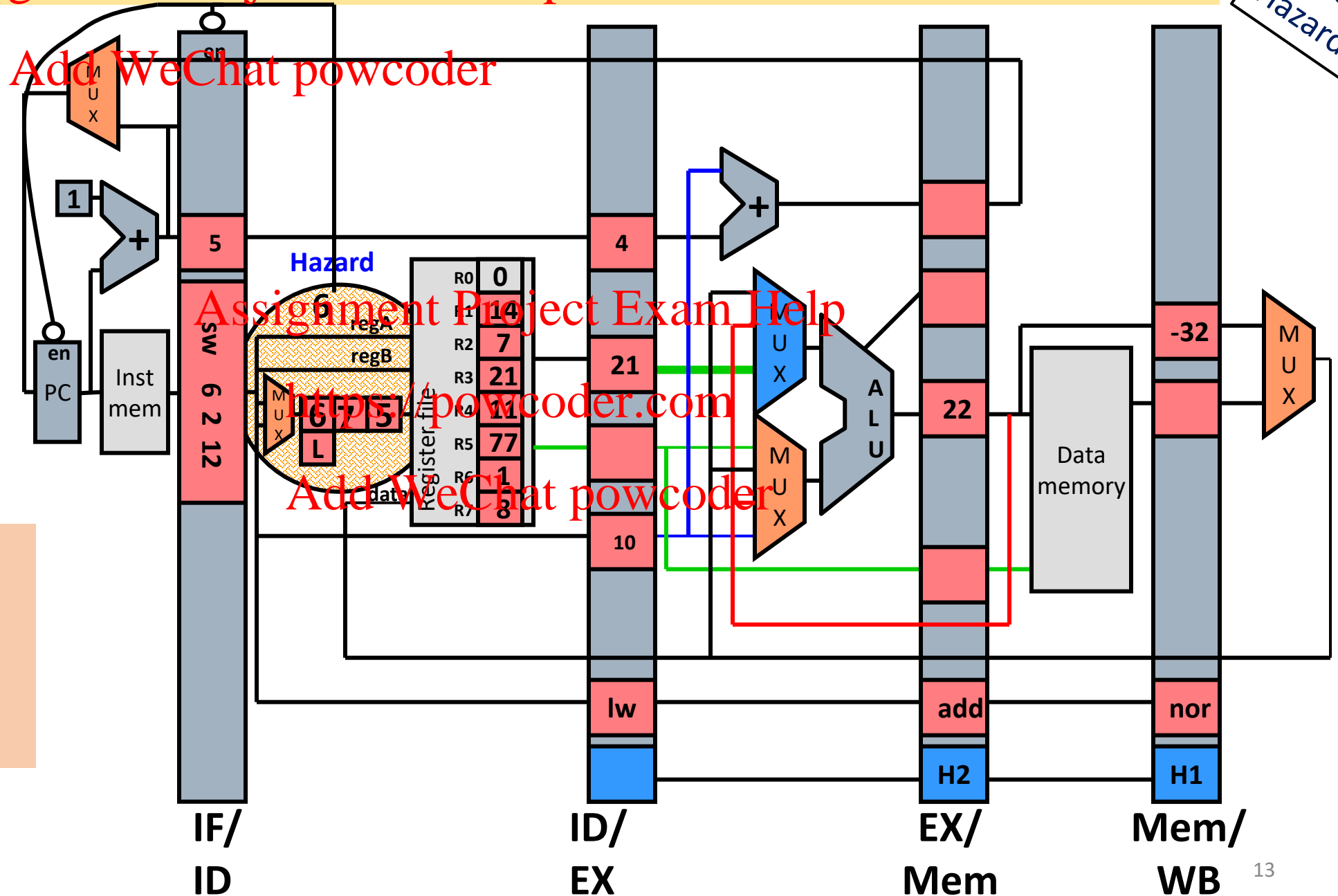
1. add 1 2 3
2. nor 3 4 5
3. add 6 3 7
4. lw 3 6 10
5. sw 6 2 12

Assignment Project Exam Help

First half of cycle 6

Add WeChat powcoder

Data Hazards

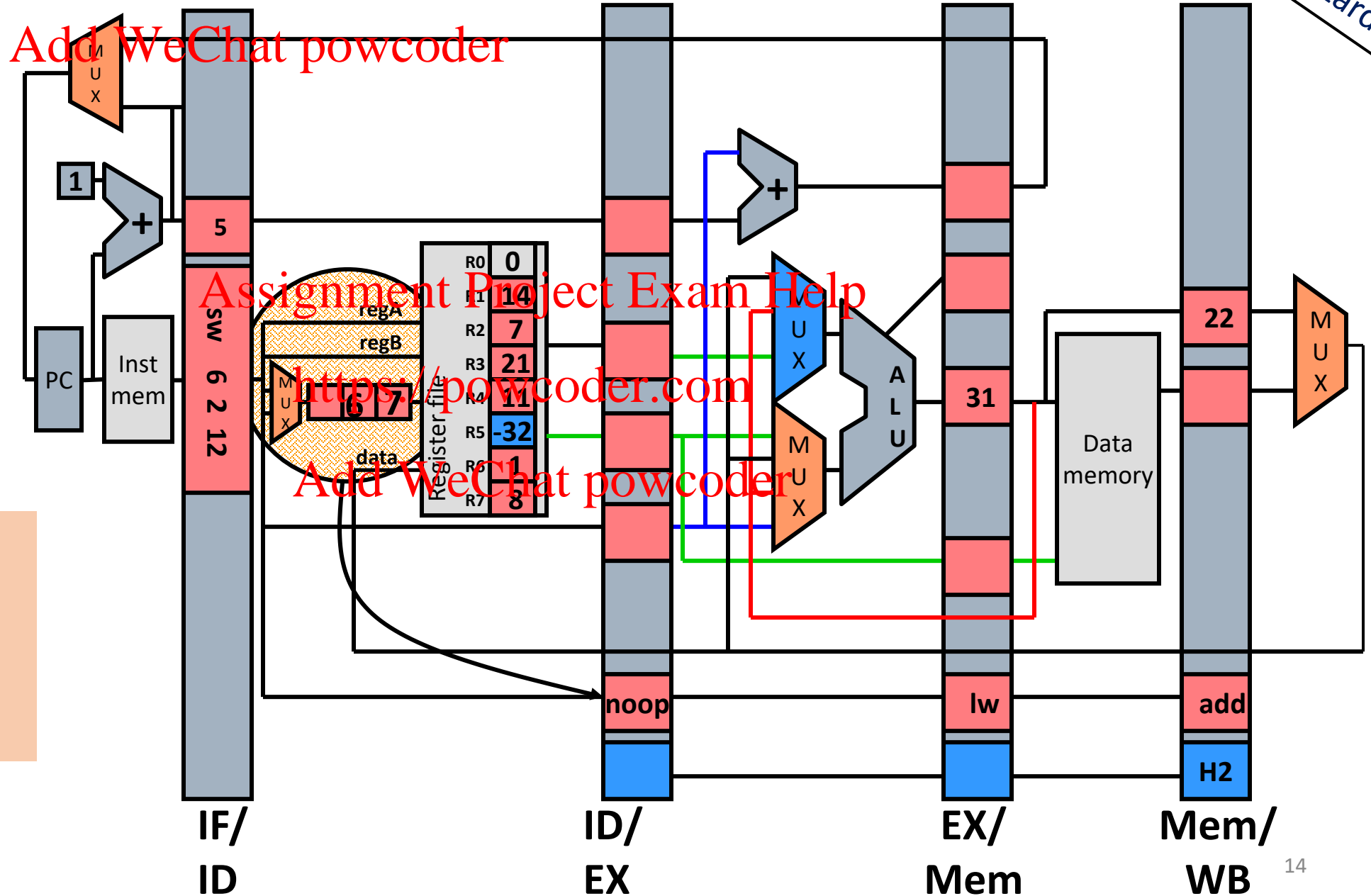


1. add 1 2 3
 2. nor 3 4 5
 3. add 6 3 7
 4. lw 3 6 10
 5. sw 6 2 12

Assignment Project Exam Help End of cycle 6

End of cycle 6

Data Hazards

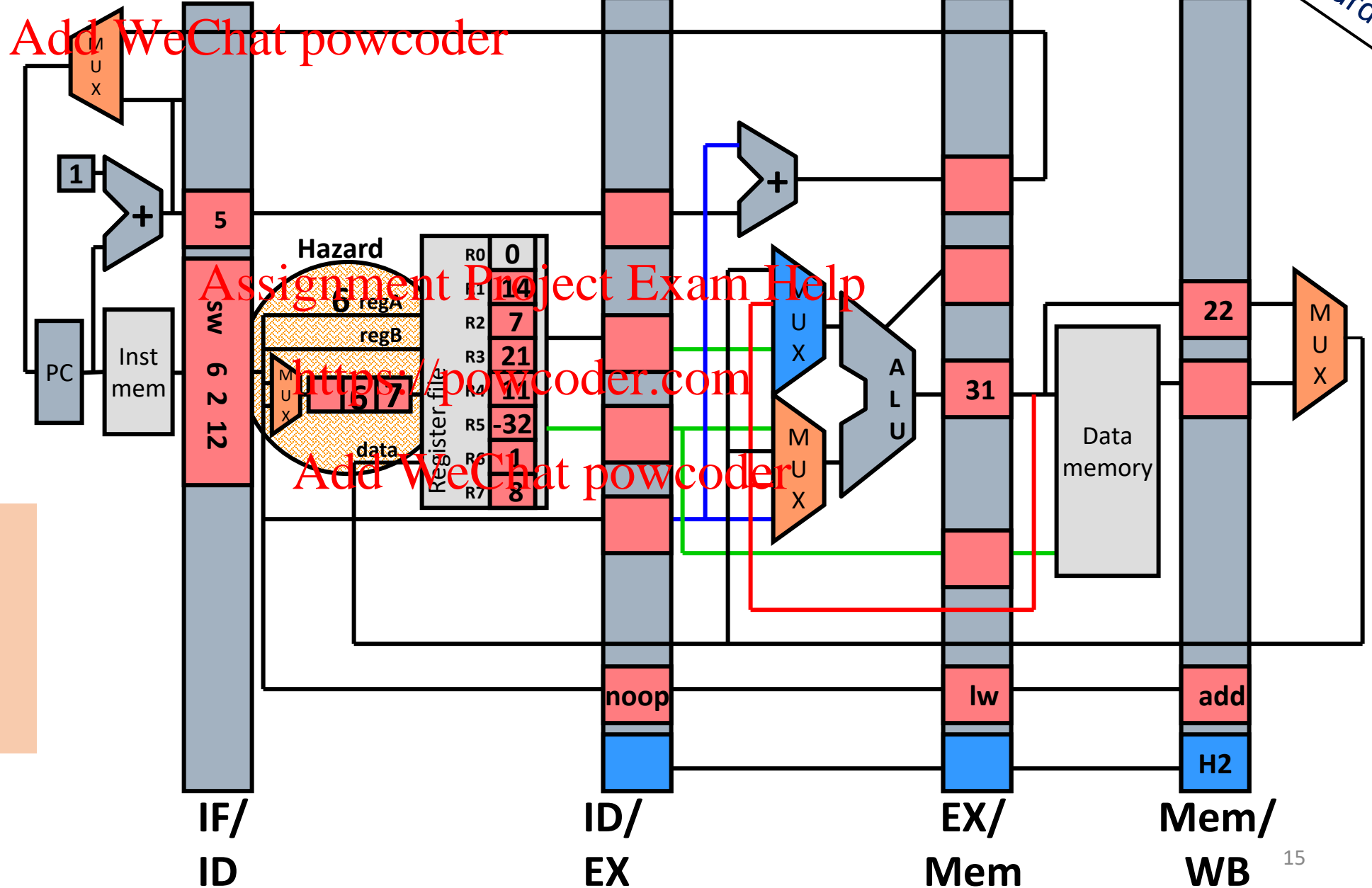


1.	add	1	2	3
2.	nor	3	4	5
3.	add	6	3	7
4.	lw	3	6	10
5.	sw	6	2	12

Assignment Project Exam Help

First half of cycle 7

Data Hazards



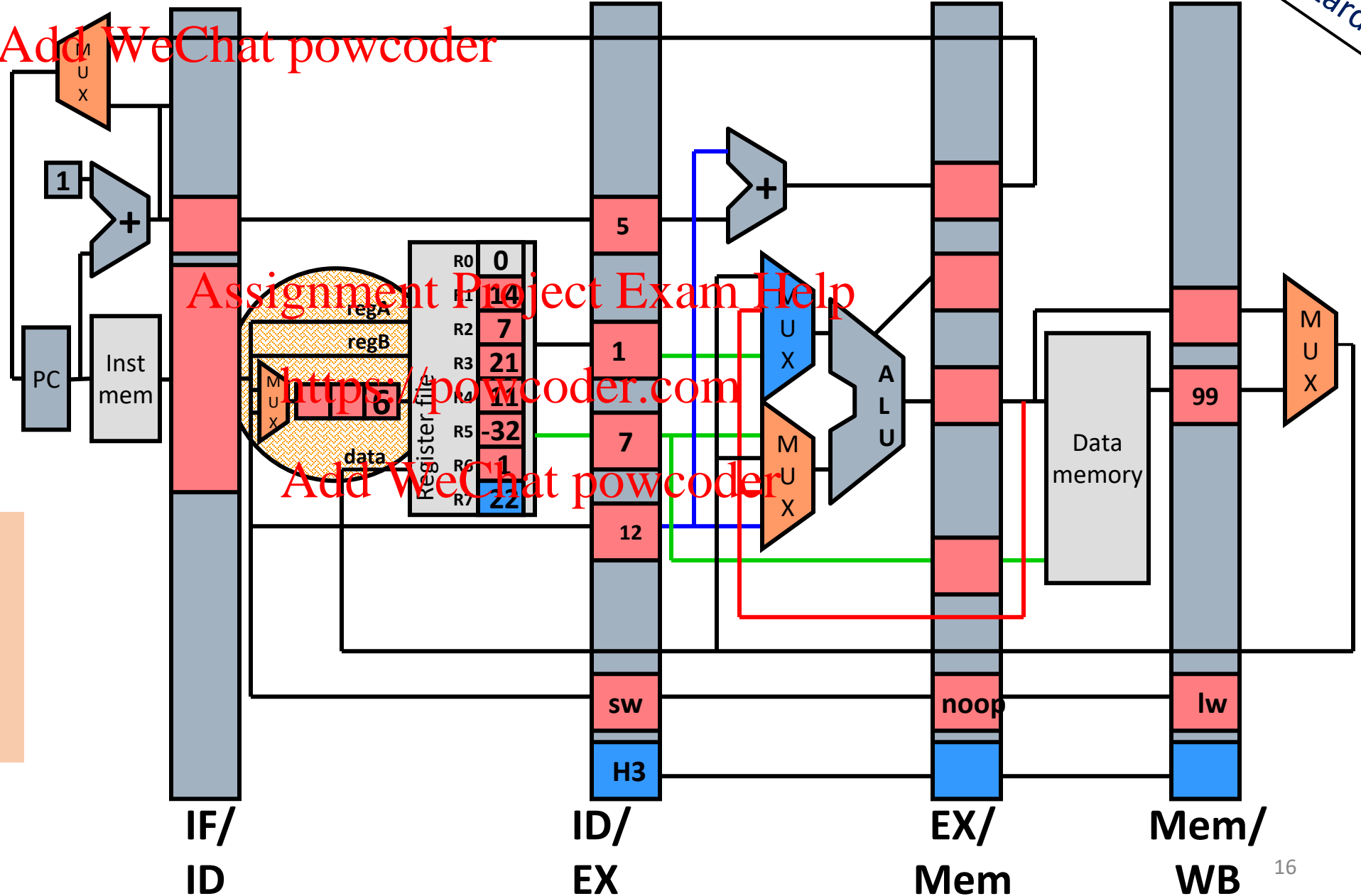
1.	add	1	2	3
2.	nor	3	4	5
3.	add	6	3	7
4.	lw	3	6	10
5.	sw	6	2	12

Assignment Project Exam Help

End of cycle 7

Data Hazards

Add WeChat powcoder



<https://powcoder.com>

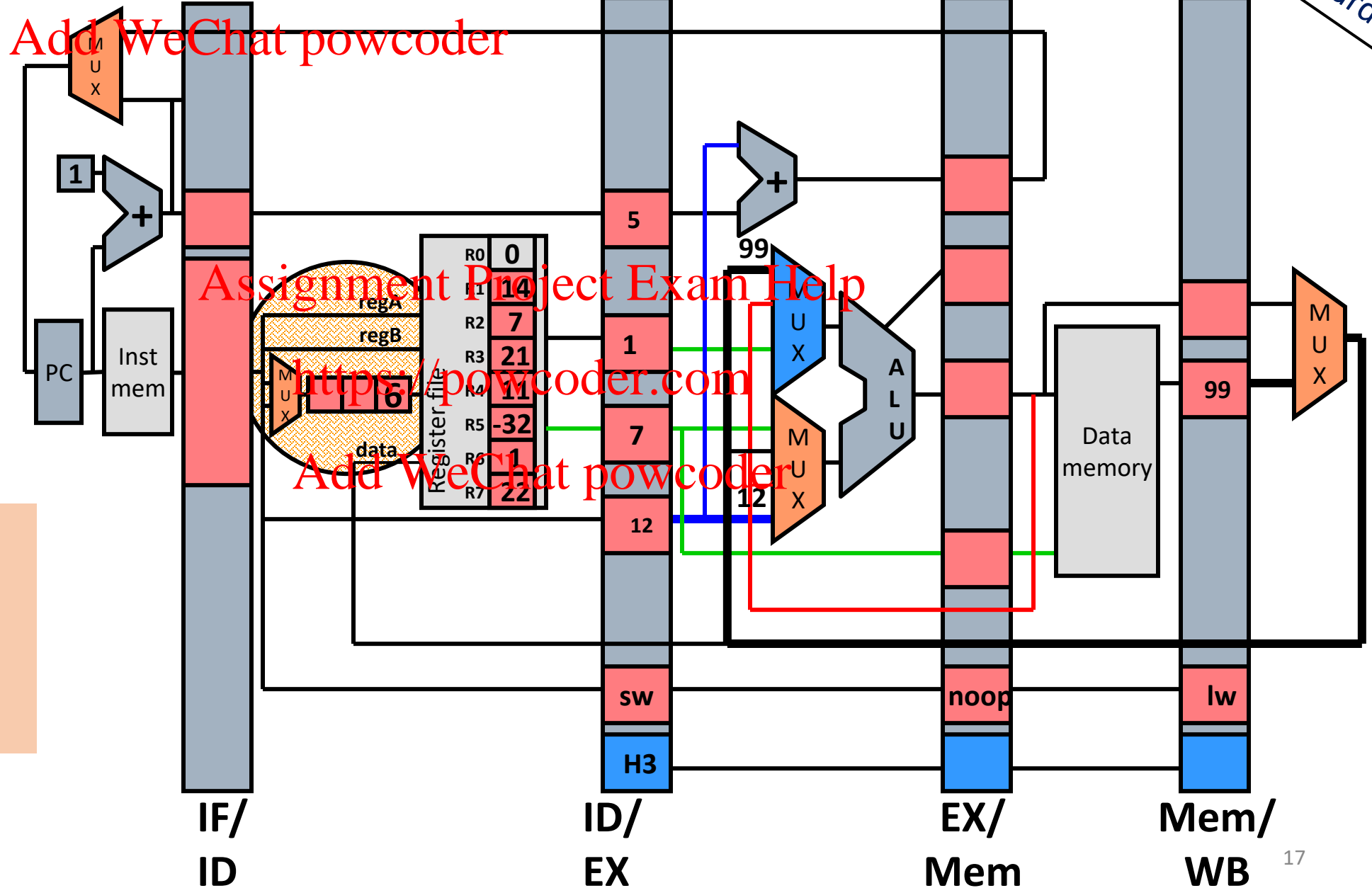
Add WeChat powcoder

1.	add	1	2	3
2.	nor	3	4	5
3.	add	6	3	7
4.	lw	3	6	10
5.	sw	6	2	12

Assignment Project Exam Help

First half of cycle 8

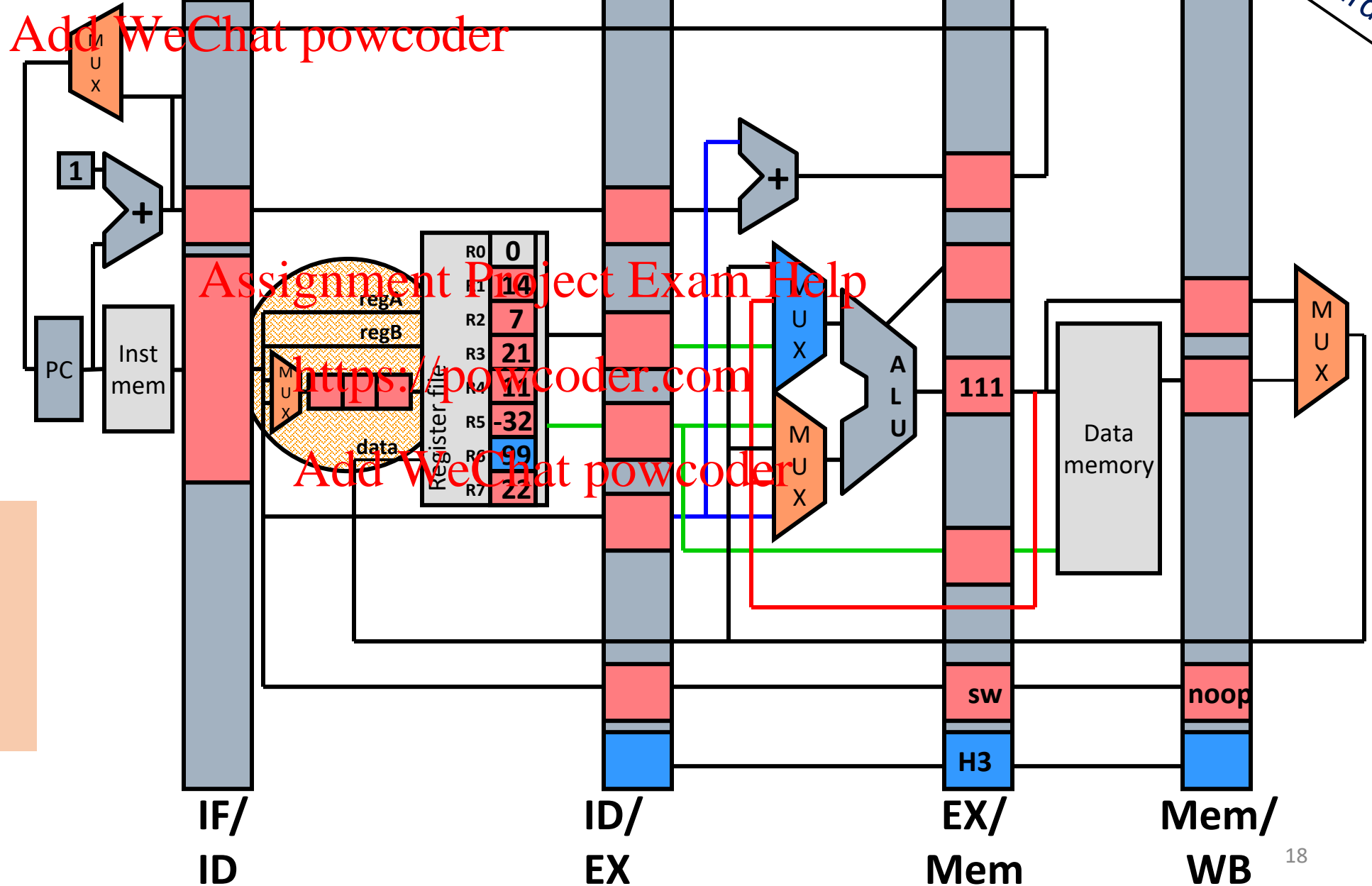
Data Hazards



Assignment Project Exam Help

End of cycle 8

Data Hazards



1.	add	1	2	3
2.	nor	3	4	5
3.	add	6	3	7
4.	lw	3	6	10
5.	sw	6	2	12

Assignment Project Exam Help

Time Graph – Detect and Forward

Add WeChat powcoder

Data Hazards

Time:	1	2	3	4	5	6	7	8	9	10	11	12	13
add 1 2 3	IF	ID	EX	ME	WB								
nor 3 4 5		IF	ID	EX	ME	WB							
add 6 3 7			IF	ID	EX	ME	WB						
lw 3 6 10				IF	ID	EX	ME	WB					
sw 6 2 12					IF	ID*	ID	EX	ME	WB			

Data forward
→

Data Forwarding – Lecture vs. Project 3



- Some questions you may have
 - What is the WBEND pipeline register in the project for?
 - Why are 3 nops required to avoid hazards in the project?
 - But only 2 nops in class?
- Answer
 - The “magic” register file
 - Lecture register file assumes internal forwarding – in a single cycle, values written to a register are immediately reflected to any reads that occur in the same cycle.
 - Project register file does not do internal forwarding.
 - Most modern processors have internal forwarding as its cheaper than having an additional pipeline register.

Assignment Project Exam Help

Project 3 Design Tips

Add WeChat powcoder



- Build up your simulator in pieces.
 - First, design code without any hazards – get the pipeline flow to work for all instructions
 - Build up your data forwarding (remember forwarding from 3 places back to EX: EX/MEM, MEM/WB, and WB/END).
 - Handle control hazards.
 - One extra cycle for stalls (simple register file implementation).
- Testcases are critical to debugging your code – don't rely on the autograder!
- Use your functionally correct previous “golden design” for testing.
- Implement without considering hazards first, test with hazard-free code, then consider hazards.
- Watch discussion videos (or attend!)



Logistics

Assignment Project Exam Help
Add WeChat powcoder


- There are 3 videos for lecture 15
 - L15_1 – Detect-and-Forward
 - L15_2 – Control-Hazards
 - L15_3 – Branch-Prediction
 - There is one worksheet for lecture 15
 1. L15 worksheet
- Assignment Project Exam Help
<https://powcoder.com>
Add WeChat powcoder

Assignment Project Exam Help

Add WeChat powcoder

L15_2 Control-Hazards

EECS 370 – Introduction to Computer Organization – Fall 2020



Assignment Project Exam Help

Learning Objectives

Add WeChat powcoder

- To identify and describe control hazards in the LC2K datapath.
- To identify the approaches to handling control hazards and their trade-offs.

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

- Resources:
 - Pipeline simulator in Resources on EECS 370 website
 - <https://vhosts.eecs.umich.edu/370simulators/pipeline/simulator.html>



Control Hazard

Assignment Project Exam Help
Add WeChat powcoder

Control hazard:

Also called branch hazard. When the proper instruction cannot execute in the proper pipeline clock cycle because the instruction that was fetched is not the one that is needed; that is, the flow of instruction addresses is not what the pipeline expected.

Assignment Project Exam Help
<https://powcoder.com>
Add WeChat powcoder

Pipeline Function for beq

Control Hazards

- Fetch: read instruction from memory.
- Decode: read source operands from registers.
- Execute: calculate target address and test for equality.
- Memory: Send target to PC if test is equal.
- Writeback: Nothing left to do.
- **Branch outcomes**
 - **Not Taken**
 - $PC = PC + 1$
 - **Taken**
 - $PC = \text{Branch Target Address}$

Control Hazards

Control Hazards

beq 1 1 10
add 3 4 5

Assignment Project Exam Help



beq fetch decode execute memory writeback

add fetch decode execute

Approaches to Handling Control Hazards

Assignment Project Exam Help

Add WeChat powcoder

I. Avoid

Make sure there are no hazards in the code.

Assignment Project Exam Help

II. Detect and stall <https://powcoder.com>

Delay fetch until branch resolved

Add WeChat powcoder

III. Speculate and Squash-if-Wrong

Go ahead and fetch more instructions in case it is correct, but stop them if they should not have been executed.

Handling Control Hazards I: Avoid all Hazards



- Do not have branch instructions!
 - Impractical.
 - Using predicated instructions works for small branches (not covered in this class)

<https://powcoder.com>

- Delay taking branch
 - Special branch instruction where next N instructions still execute, and then branches after those
 - Not covered in more detail in this class

Handling Control Hazards II:

Detect and Stall

- Detection

- Must wait until decode
- Compare opcode to segment register
- Alternately, this is just another control signal

<https://powcoder.com>

- Stall

- Keep current instructions in fetch
- Insert noops
- Pass noop to decode stage, not execute!

Assignment Project Exam Help

Control Hazards - Stall

Add WeChat powcoder

beq 1 1 10
add 3 4 5

Assignment Project Exam Help



beq fetch decode execute memory writeback

Add WeChat powcoder

add fetch* fetch* fetch* fetch

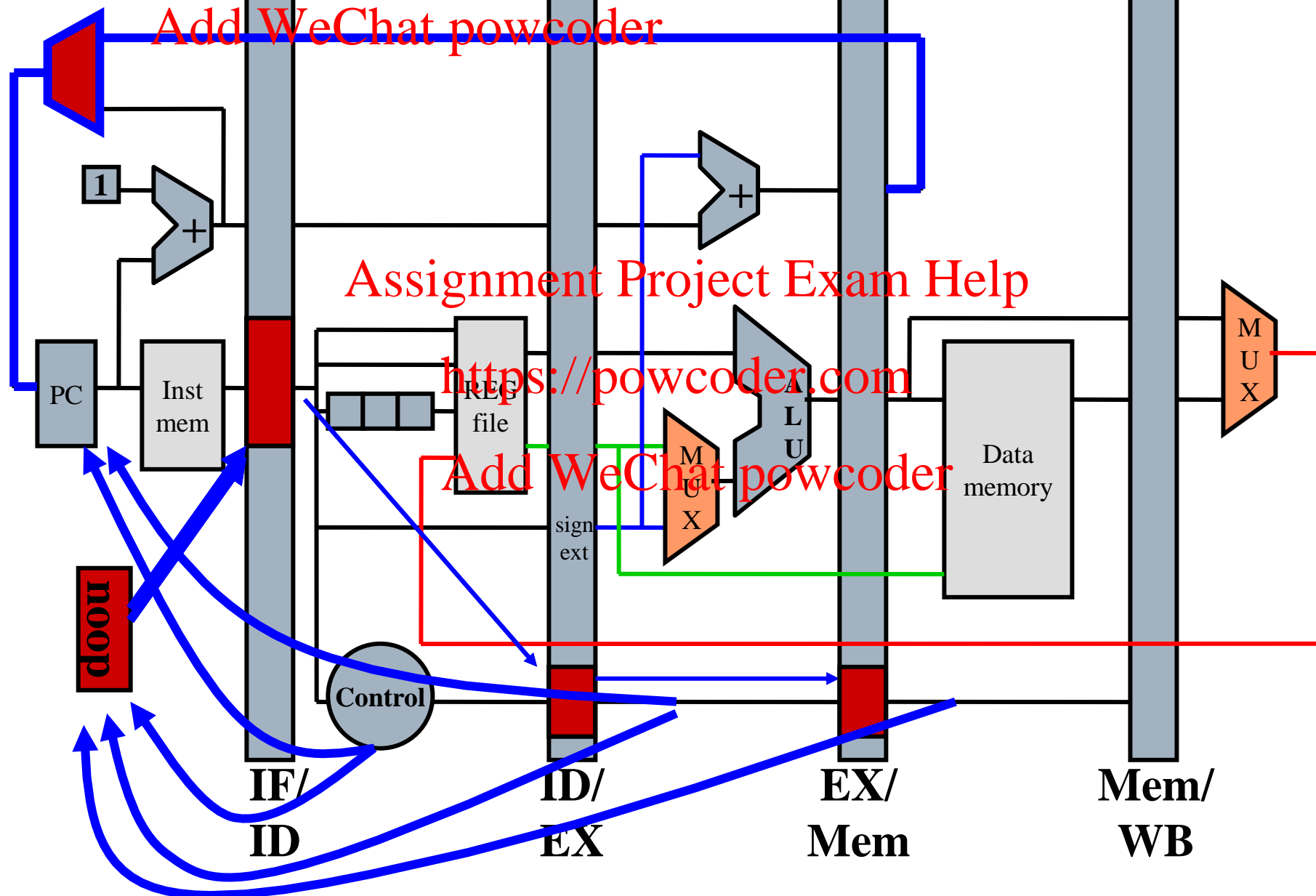
OR

Branch target
address fetch

Assignment Project Exam Help

Control Hazards

Add WeChat powcoder



Problems with Detect and Stall

- CPI increases every time a branch is detected!
- Is that necessary? Not always!
 - Branch not always taken
 - Let us assume that it is NOT taken...
 - In this case, we can ignore the beq (treat it like a noop).
 - Keep fetching PC + 1
 - What if we are wrong?
 - OK, as long as we do not COMPLETE any instructions we mistakenly executed.
 - i.e., make changes that will be seen later such as changing register or memory values.

Handling Control Hazards III: Speculate and Squash

- Speculate: assume not equal
 - Keep fetching from PC+1 until we know that the branch is really taken.
- Squash: stop bad instructions if taken
 - Send a noop to Decode, Execute, and Memory
 - Send target address to PC

Problems with Fetching PC+1



- CPI increases every time a branch is taken!
 - About 50%-66% of time
 - Is that necessary?

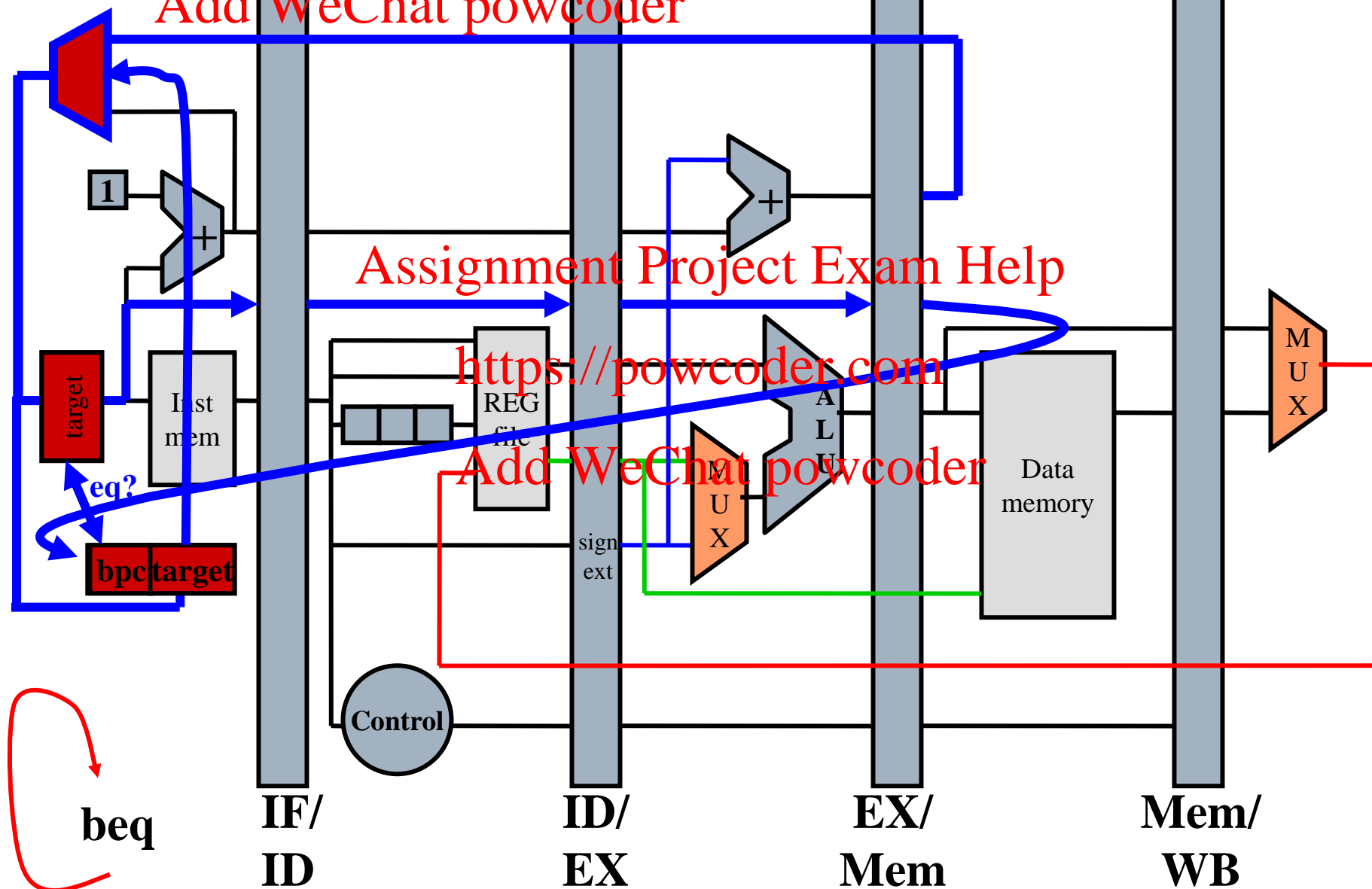
<https://powcoder.com>

No! But how can you fetch from the target before you even know the previous instruction is a branch – much less whether it is taken?

Assignment Project Exam Help

Add WeChat powcoder

Control Hazards





Logistics

Assignment Project Exam Help
Add WeChat powcoder


- There are 3 videos for lecture 15
 - L15_1 – Detect-and-Forward
 - L15_2 – Control-Hazards
 - L15_3 – Branch-Prediction
 - There is one worksheet for lecture 15
 1. L15 worksheet
- Assignment Project Exam Help
<https://powcoder.com>
Add WeChat powcoder

Assignment Project Exam Help

Add WeChat powcoder

L15_3 Branch-Prediction

EECS 370 – Introduction to Computer Organization – Fall 2020



Assignment Project Exam Help

Learning Objectives

Add WeChat powcoder

- To identify the control and state for simple branch prediction.
- Ability to understand metrics for branch prediction accuracy.

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

Branch Prediction

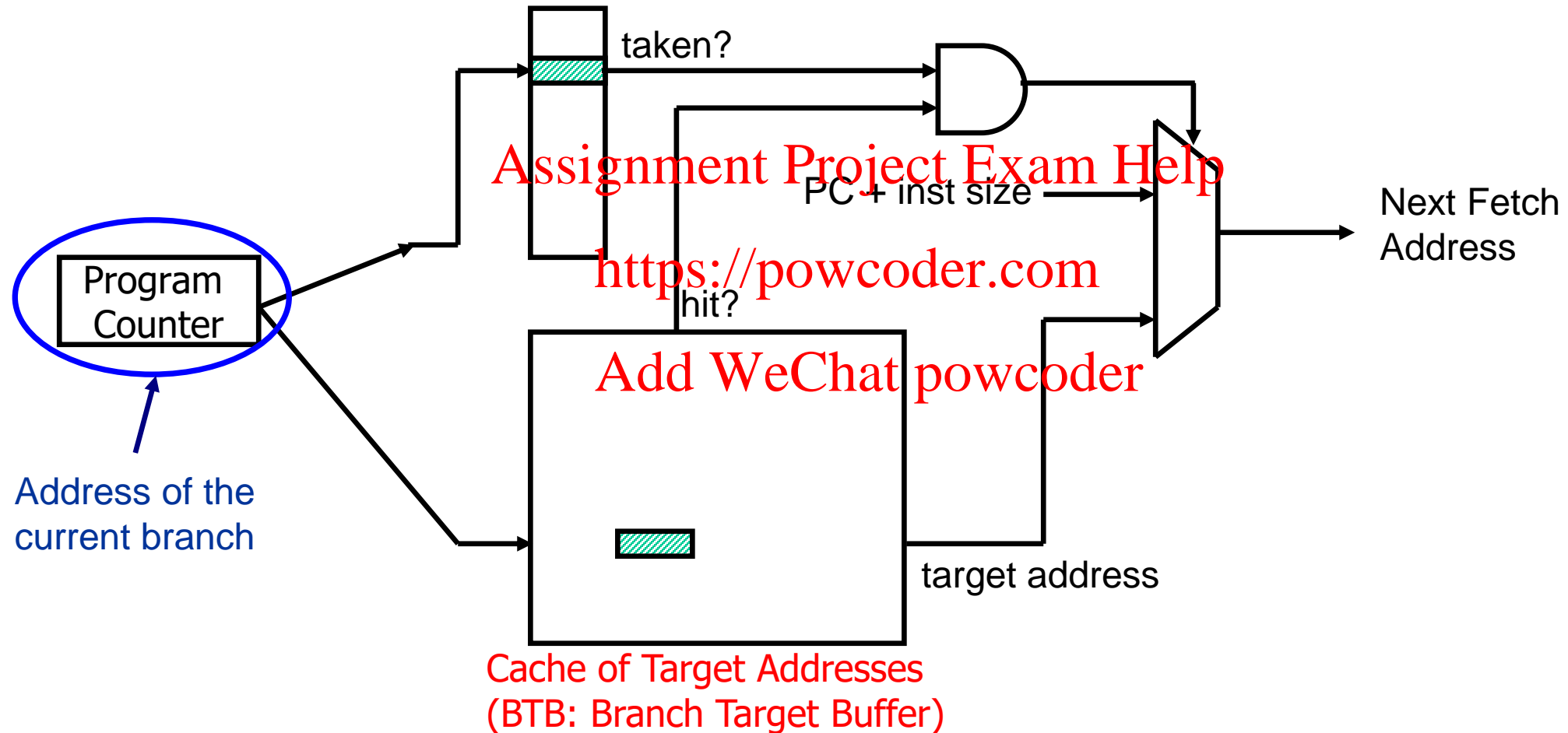


- Predict the next fetch address (to be used in the next cycle)
- Requires three things to be predicted at fetch stage:
 - Whether the fetched instruction is a branch
 - Branch direction (if conditional)
 - Branch target address (if direction is taken)
- Observation: Target address remains the same for a conditional direct branch across dynamic instances
 - Store the target address from previous instance and access it with the PC
 - Called Branch Target Buffer (BTB) or Branch Target Address Cache

Fetch Stage with Branch Prediction

Branch
Prediction

Direction predictor (2-bit counters)



Branch Direction Prediction



- "Branch direction" refers to whether the branch was taken or not
- Two methods for predicting direction:
 - Static - We predict once during compilation, and that prediction never changes
 - Dynamic - We predict (potentially) many times during execution, and the prediction may change over time
- *Static vs dynamic strategies are a very common topic in computer architecture*

Branch Direction Prediction



- Compile time (static)

- Always not taken
- Always taken
- BTFN (Backward taken, forward not taken)
- Profile based (likely direction)
- Program analysis based (likely direction)

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

- Run time (dynamic)

- Last time prediction (single-bit)
- Two-bit counter based prediction
- Two-level prediction (global vs. local)
- Hybrid

Branch Direction Prediction (Static)



- Always not-taken
 - Simple to implement: no need for BTB, no direction prediction
 - Low accuracy: ~30-40%
 - Compiler can layout code such that the likely path is the "not taken" path
- Always taken
 - No direction prediction
 - Better accuracy: ~60-70%
 - Backward branches (i.e. loop branches) are usually taken
 - Backward branch: target address lower than branch PC
- Backward taken, forward not taken (BTFN)
 - Predict backward (loop) branches as taken, others not-taken

<https://powcoder.com>

Add WeChat powcoder

Branch Direction Prediction (Dynamic)



- Last time predictor

- Single bit per branch (stored in BTB)
- Indicates which direction branch went last time it executed

TTTTTTTTTTNNNNNNNNNN → 90% accuracy

Assignment Project Exam Help

<https://powcoder.com>

- Always mispredicts the last iteration and the first iteration of a loop branch
 - Accuracy for a loop with N iterations = $(N-2)/N$

Add WeChat powcoder

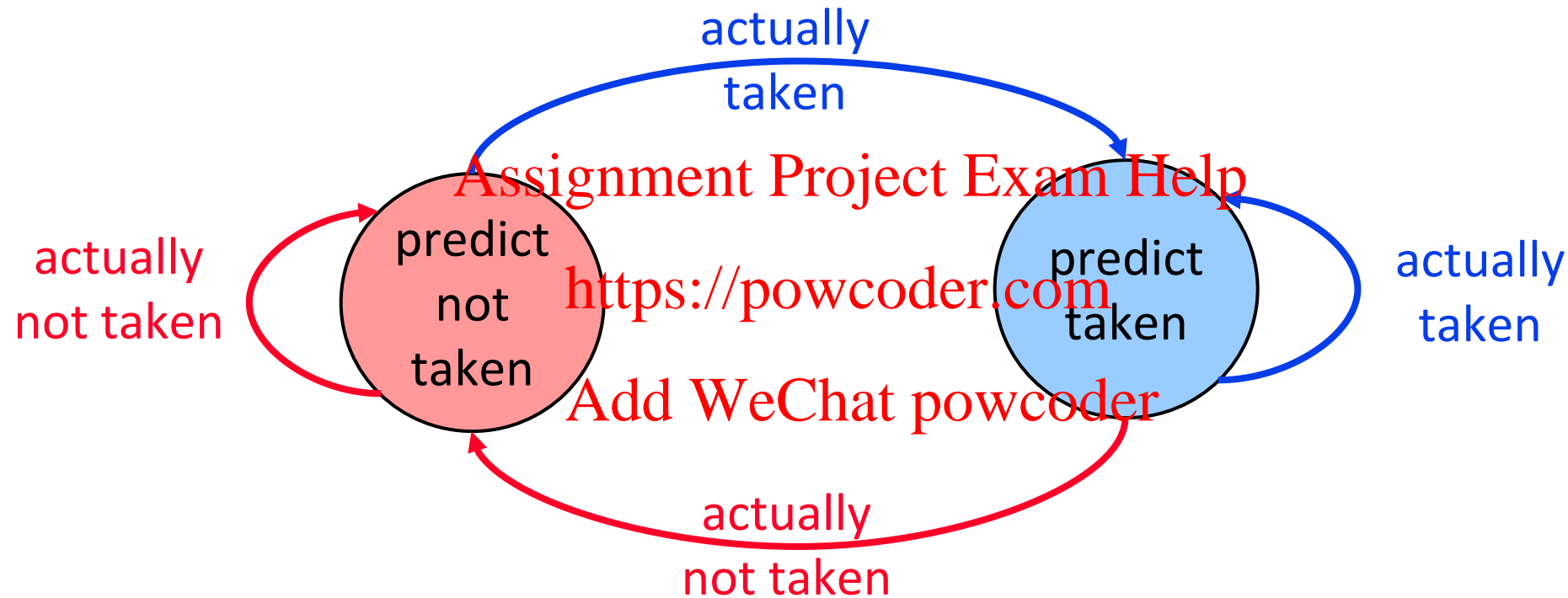
+ Loop branches for loops with large number of iterations

-- Loop branches for loops will small number of iterations

TNTNTNTNTNTNTNTN → 0% accuracy

State Machine for Last-Time Prediction

Branch
Prediction



Improving the Last Time Predictor



- Problem: A last-time predictor changes its prediction from T \rightarrow NT or NT \rightarrow T too quickly
 - Even though the branch may be mostly taken or mostly not taken
- Solution Idea: Add hysteresis to the predictor so that prediction does not change on a single different outcome
 - Use two bits to track the history of predictions for a branch instead of a single bit
 - Can have 2 states for T or NT instead of 1 state for each

<https://powcoder.com>

Two-Bit Counter Based Prediction

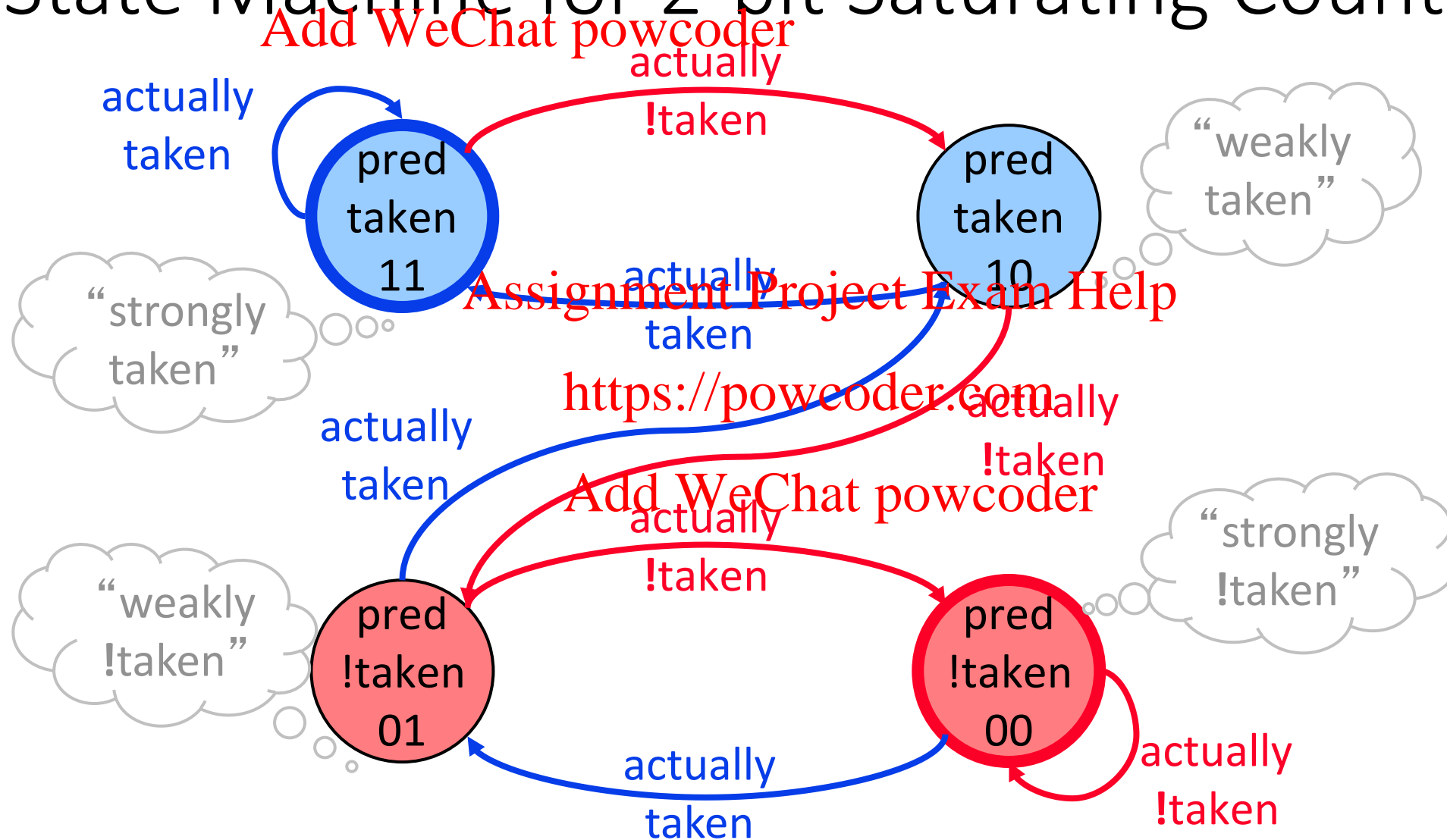
- Each branch associated with a two-bit counter
- One more bit provides hysteresis
- A strong prediction does not change with one single different outcome

<https://powcoder.com>

- + Better prediction accuracy
- More hardware cost (but counter can be part of a BTB entry)

State Machine for 2-bit Saturating Counter

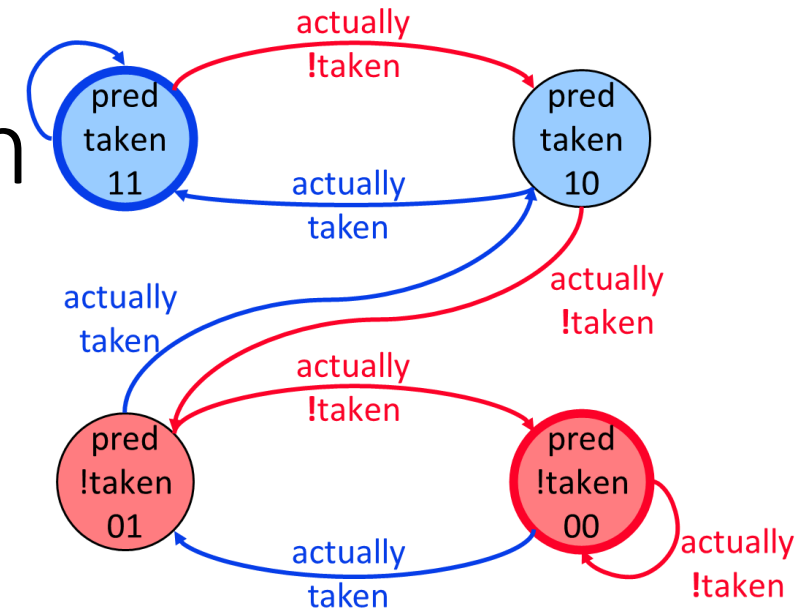
Branch
Prediction



Two-Bit Counter Based Prediction

- What is the prediction accuracy of a branch with the following sequence of taken/not taken evaluations

T T T T N T N N N N T N T N N



<https://powcoder.com>

Branch	T	T	T	T	N	T	T	N	N	N	T	N	T	N	N
State	10	11	11	11	X	10	11	X	X	01	X	01	X	01	00
Pred	T	T	T	T	T	T	T	T	T	N	N	N	N	N	N

Branch Prediction

Assignment Project Exam Help
Add WeChat powcoder



- Predict not taken: ~50% accurate.
- Predict backward taken: ~65% accurate.
- Predict same as last time: ~80% accurate.

<https://powcoder.com>

- Pentium: ~85% accurate.
- Pentium Pro: ~92% accurate.
- Best paper designs: ~96% accurate.

Can We Do Better? – Beyond EECS 370

Branch
Prediction

- Absolutely – take EECS 470
- TAGE Branch Prediction
 - [Tagged GEometric length https://powcoder.com](https://powcoder.com)
- [Neural Branch Prediction](#)
- State of branch prediction
 - [Branch Prediction is Not a Solved Problem](#)

Logistics

Assignment Project Exam Help
Add WeChat powcoder

Branch
Prediction

- There are 3 videos for lecture 15
 - L15_1 – Detect-and-Forward
 - L15_2 – Control-Hazards
 - L15_3 – Branch-Prediction
- There is one worksheet for lecture 15
 1. L15 worksheet

<https://powcoder.com>

Add WeChat powcoder