L12_1 Performance_Metrics

EECS 370 – Introduction to Computer Organization – Fall 2020

Reminder

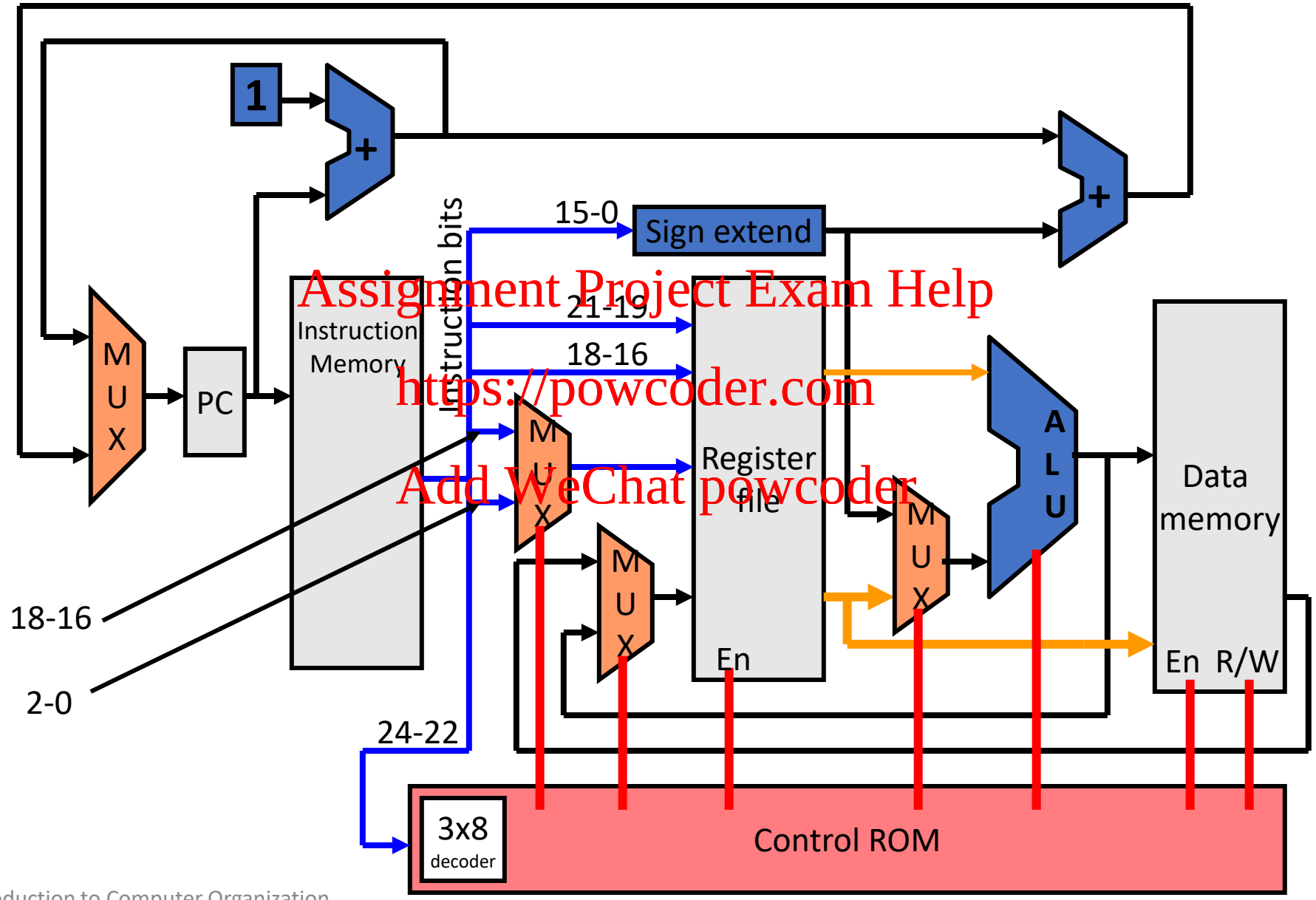
- Midterm is on 10/20
- Leverage past exams
  - There are several available in the Exams tab on the website
  - Start *now* with the questions for topics we already know
  - Next week take an exam
    - Set a timer
    - Complete the exam without looking at the solutions
    - Check your answers
- There will be an exam review session

Learning Objectives

- To understand and be able to apply performance metrics to evaluate processor performance.

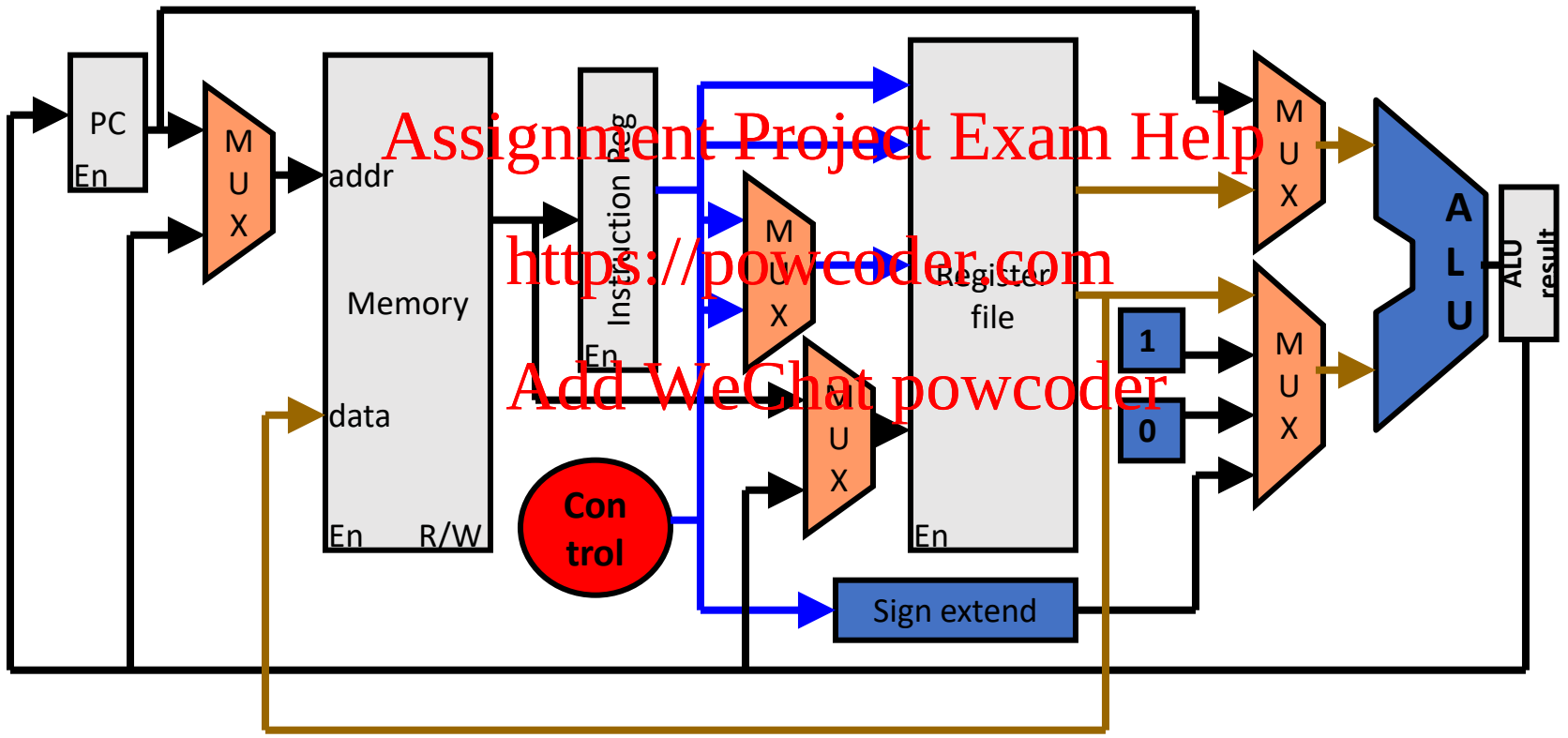
Single-cycle LC2K Datapath

Single-Cycle Review



Multicycle LC2K Datapath

Multi-Cycle Review



Riddle #1

Problem: How long does it take to help 10 customers?

- You work at McDonald's by yourself
- You help one customer at a time

Latencies:
30 s – time to take order
30 s – time for customer to pay
60 s – prepare food and give to customer
0 s – everything else

Total Latency: 2 minutes

10 * 2 min = 20 minutes

Single and Multicycle Performance

Single/Multi-Cycle Review

Latencies:
1 ns – Register File read/write time
2 ns – ALU/adder
2 ns – memory access
0 ns – MUX, PC access, sign extend, ROM

1. Assuming the above delays, what is the best cycle time that the LC2K single and multi-cycle datapath could achieve?

2. Assuming the above delays, for a program consisting of 25 LW, 10 SW, 45 ADD, and 20 BEQ, which is faster?

Single and Multicycle Performance

Single/Multi-Cycle Review

Latencies:
1 ns – Register File read/write time
2 ns – ALU/adder
2 ns – memory access
0 ns – MUX, PC access, sign extend, ROM

1. Assuming the above delays, what is the best cycle time that the LC2K single and multi-cycle datapath could achieve?

SC: 2 + 1 + 2 + 2 + 1 = 8 ns
MC: MAX(2, 1, 2, 2, 1) = 2ns

2. Assuming the above delays, for a program consisting of 25 LW, 10 SW, 45 ADD, and 20 BEQ, which is faster?

SC: 100 cycles * 8 ns = 800 ns
MC: (25*5 + 10*4 + 45*4 + 20*4)cycles * 2ns = 850 ns

What good is multi-cycle?

Single and Multicycle Performance

Single/Multi-Cycle Review

Latencies:
2 ns – Register File read/write time
2 ns – ALU/adder
2 ns – memory access
0 ns – MUX, PC access, sign extend, ROM

1. What if the register file access is increased to 2ns, does that change the answer to the previous question?

SC: 2 + 2 + 2 + 2 + 2 = 10 ns
MC: MAX(2, 2, 2, 2, 2) = 2ns

2. Assuming the above delays, for a program consisting of 25 LW, 10 SW, 45 ADD, and 20 BEQ, which is faster?

SC: 100 cycles * 10 ns = 1000 ns
MC: (25*5 + 10*4 + 45*4 + 20*4)cycles * 2ns = 850 ns

Balancing delays helps multi-cycle

Single-Cycle Performance

Latencies:
1 ns – Register File read/write time
2 ns – ALU/adder
2 ns – memory read access
20 ns – memory **write** access
0 ns – MUX, PC access, sign extend, ROM

1. Assuming the above delays, what is the best cycle time that the LC2K single-cycle datapath could achieve?

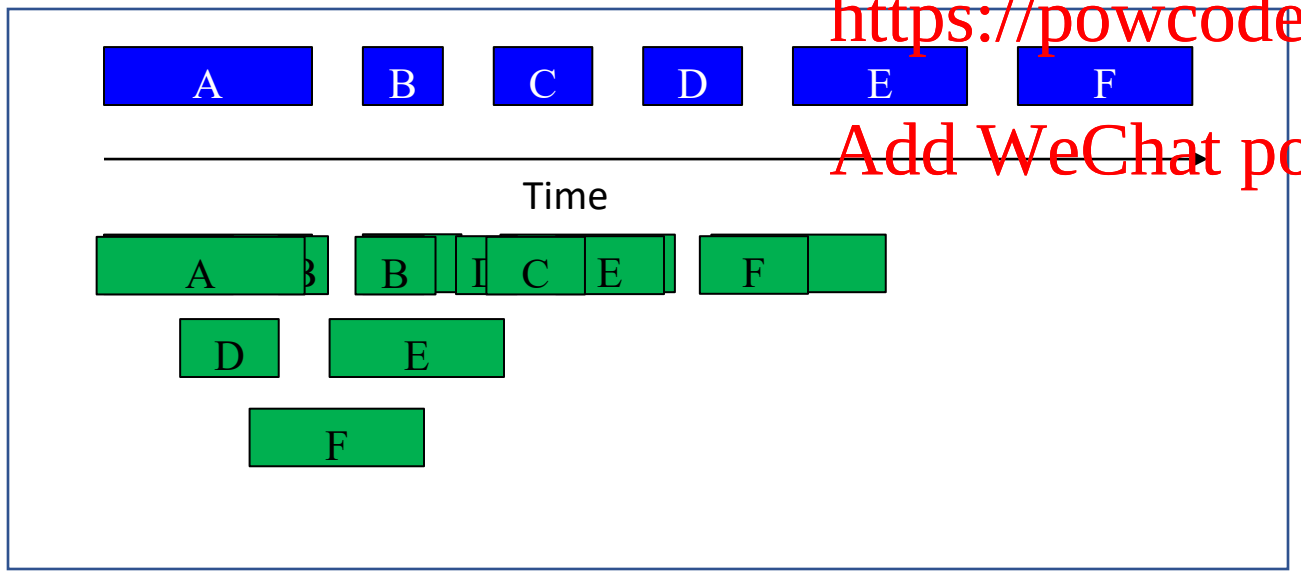
```
lw: 2 + 1 + 2 + 2 + 1 = 8 ns

add: 2 + 1 + 2 + 1 = 6 ns

sw: 2 + 1 + 2 + 20 = 25 ns
```

Strategies to Execute Programs Faster

- Eliminate operations – e.g., better algorithm
- Decrease operation latency – e.g., smaller transistors, faster clock
- Execute operations in parallel – e.g., parallel execution



Performance Metrics

1. **Response time**: when is my job done (time)?
   - When will my books arrive from amazon.com?
   - How long will this program/instruction take?

2. **Throughput**: how much work can get done within a specified time (work/time)?
   - How many books will amazon.com sell this week?
   - How many programs/instructions complete per hour?
   - Improved relatively easily by using multiprocessors.

Performance Metrics – Execution Time

- Response time for a program is its **execution time**

Execution time (for an application):

= total instructions executed x CPI x clock period

- **Called the "Iron Law" of performance**

- CPI = Cycles Per Instruction = avg number of clock cycles per instruction *for an application*

- For multi-cycle processor implementations we need:
  - Cycles necessary for each type of instruction
  - Mix of instructions executed in the application (dynamic instruction execution profile)

Performance Metrics - Units

- What are the units of (instructions executed x CPI x clock period)?

$$\text{instr.} \times \frac{\text{cycle}}{\text{instr.}} \times \frac{\text{time}}{\text{cycle}} = \text{time}$$

How Far Have We Come?

Performance Metrics

Latencies:
2 ns – Register File read/write time
2 ns – ALU/adder
2 ns – memory access
0 ns – MUX, PC access, sign extend, ROM

2. Assuming the above delays, for a program consisting of 25 LW, 10 SW, 45 ADD, and 20 BEQ, which is faster?

- Single-cycle processor implementations
  - CPI = ?
  - clock period = ?
- Multi-cycle processor implementations
  - CPI = ?
  - clock period = ?
- Next step: improve CPI without impacting clock period
  - The easiest thing to do is to work on multiple instructions at the same time.

How Far Have We Come?

Performance Metrics

Latencies:
2 ns – Register File read/write time
2 ns – ALU/adder
2 ns – memory access
0 ns – MUX, PC access, sign extend, ROM

2. Assuming the above delays, for a program consisting of 25 LW, 10 SW, 45 ADD, and 20 BEQ, which is faster?

- Single-cycle processor implementations
  - CPI = 1
  - clock period = 10 ns
  - "Iron law" execution time = instr. * CPI * clock period = 100 * 1 * 10ns = 1000 ns

- Multi-cycle processor implementations
  - CPI = 4.25
  - clock period = 2 ns
  - "Iron law" execution time = instr. * CPI * clock period = 100 * 4.25 * 2ns = 850 ns

- Next step: improve CPI without impacting clock period
  - The easiest thing to do is to work on multiple instructions at the same time.

Logistics

- There are 3 videos for lecture 12
  - L12_1 – Performance_Metrics
  - L12_2 – Pipelining_Introduction
  - L12_3 – Pipelining_Execution-Example
- There is one worksheet for lecture 12
  1. L12 worksheet

L12_2 Pipelining-Introduction

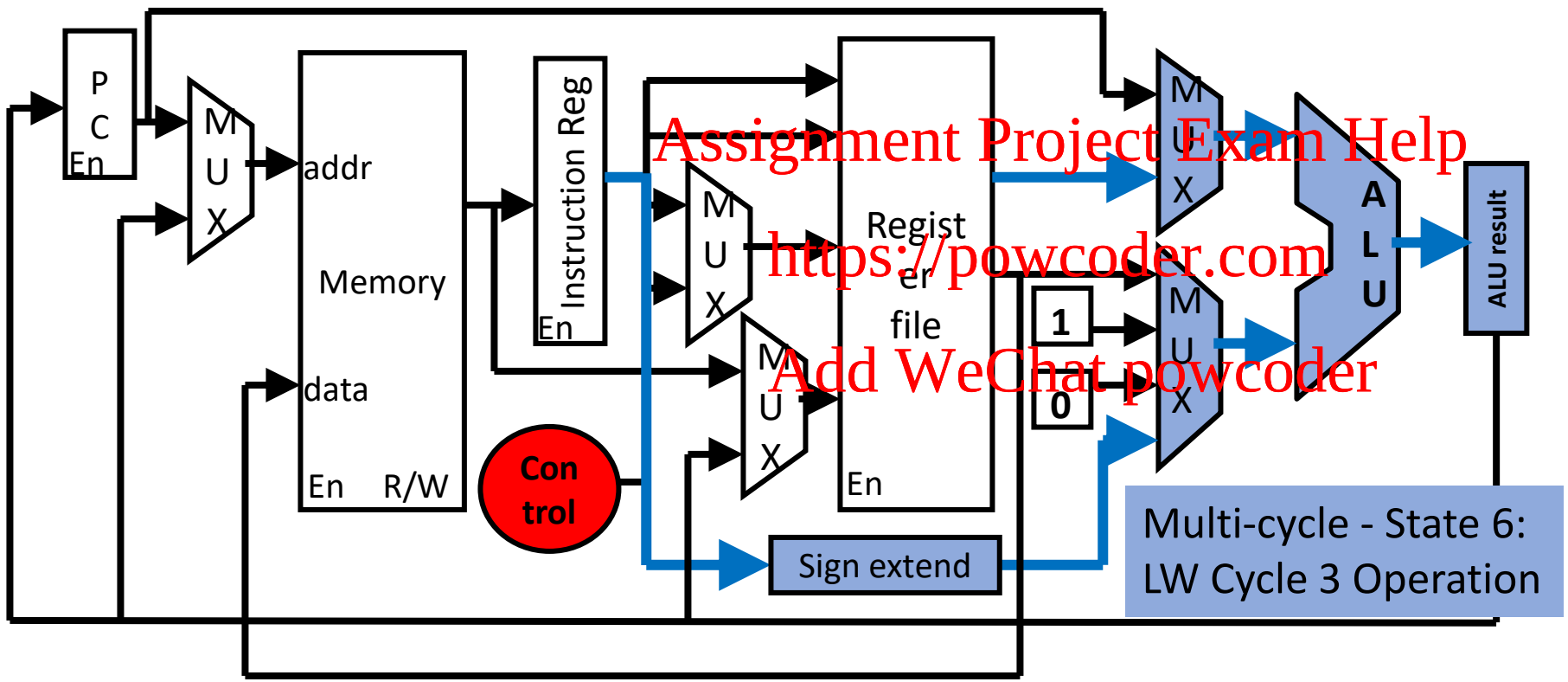
EECS 370 – Introduction to Computer Organization – Fall 2020

Learning Objectives

- To describe the overlap in executing instructions, i.e., executing more than one instruction at any one time in a datapath.
  - We want to see how to overlap instructions – not wait for one to finish before starting the next.
- To identify the stages of a pipeline datapath and describe the dataflow of instruction execution.
  - What control, state, and execution units are necessary to support overlap of instruction execution, and how does data flow between these units.

Utilization

Multi-Cycle Review



Multi-cycle - State 6: LW Cycle 3 Operation

Observation: at any time, utilization is low for single and multi-cycle datapaths

Optimization: start next instruction before current instruction finishes execution.

Riddle #2

Problem: How long does it take to help 10 customers?

- Abigail, Jack and Yujia work together at Burger King
  - Abigail takes the order from the customer
  - Jack handles the cash register
  - Yujia makes the food and gives it to the customer
  - Abigail starts helping the next customer immediately after taking the order of the first

Latencies:
30 s – time to take order
30 s – time for customer to pay
60 s – prepare food and give to customer
0 s – everything else

Latency for first customer: 2 minutes
Latency for each customer after: 1 min.
Total time: 11 minutes

Pipelining

Want to execute an instruction?

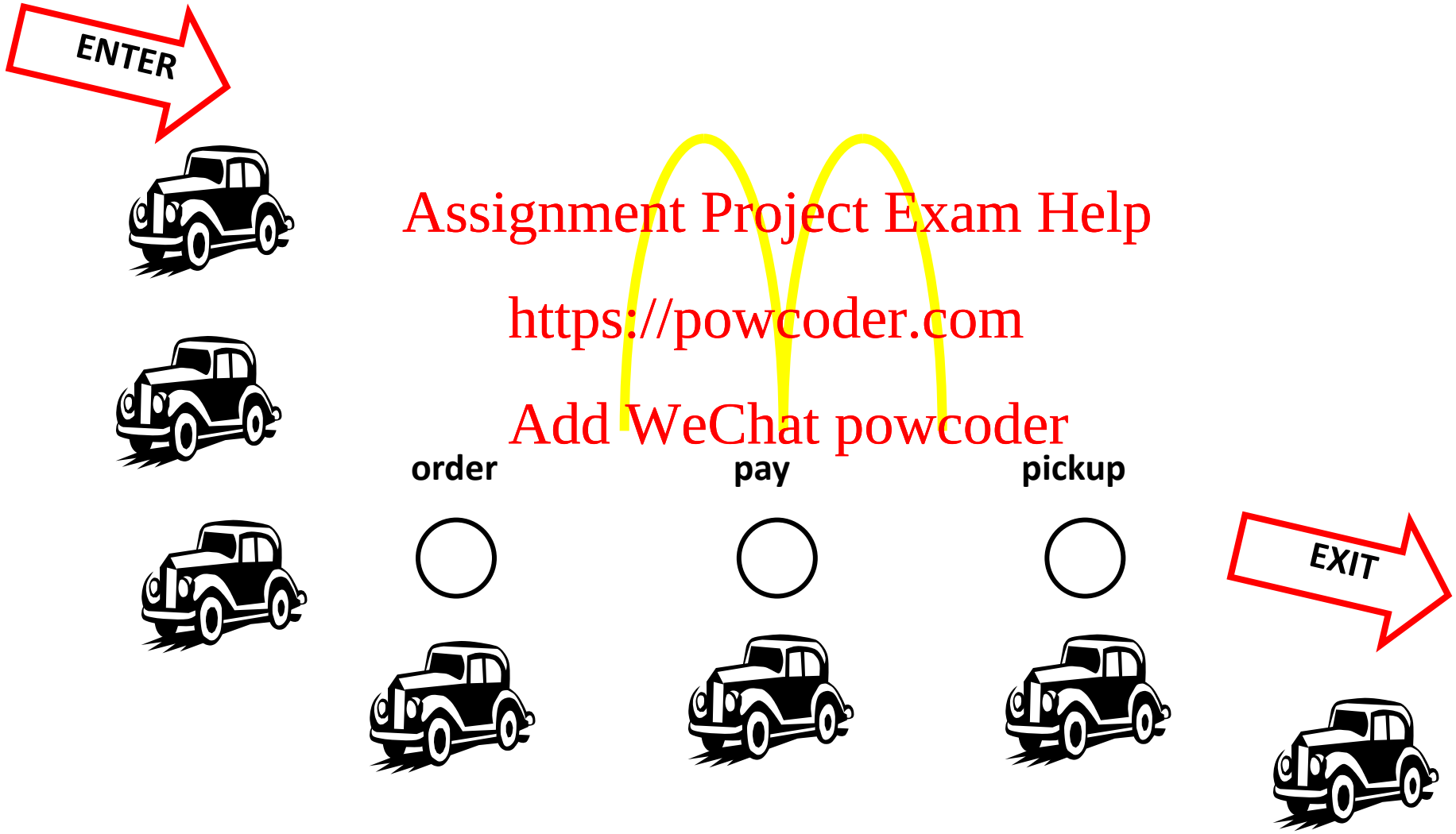
- Build a processor (multi-cycle)
- Find instructions
- Line up instructions (1, 2, 3, …)
- Overlap execution
  - Cycle #1: Fetch 1
  - Cycle #2: Fetch 2 Decode 1
  - Cycle #3: Fetch 3 Decode 2 ALU 1
  - . . . . . .
- This is called *pipelining* instruction execution.
- Used extensively for the first time on IBM 360 (1960s).
- CPI approaches 1

Pipelining – Automotive Assembly

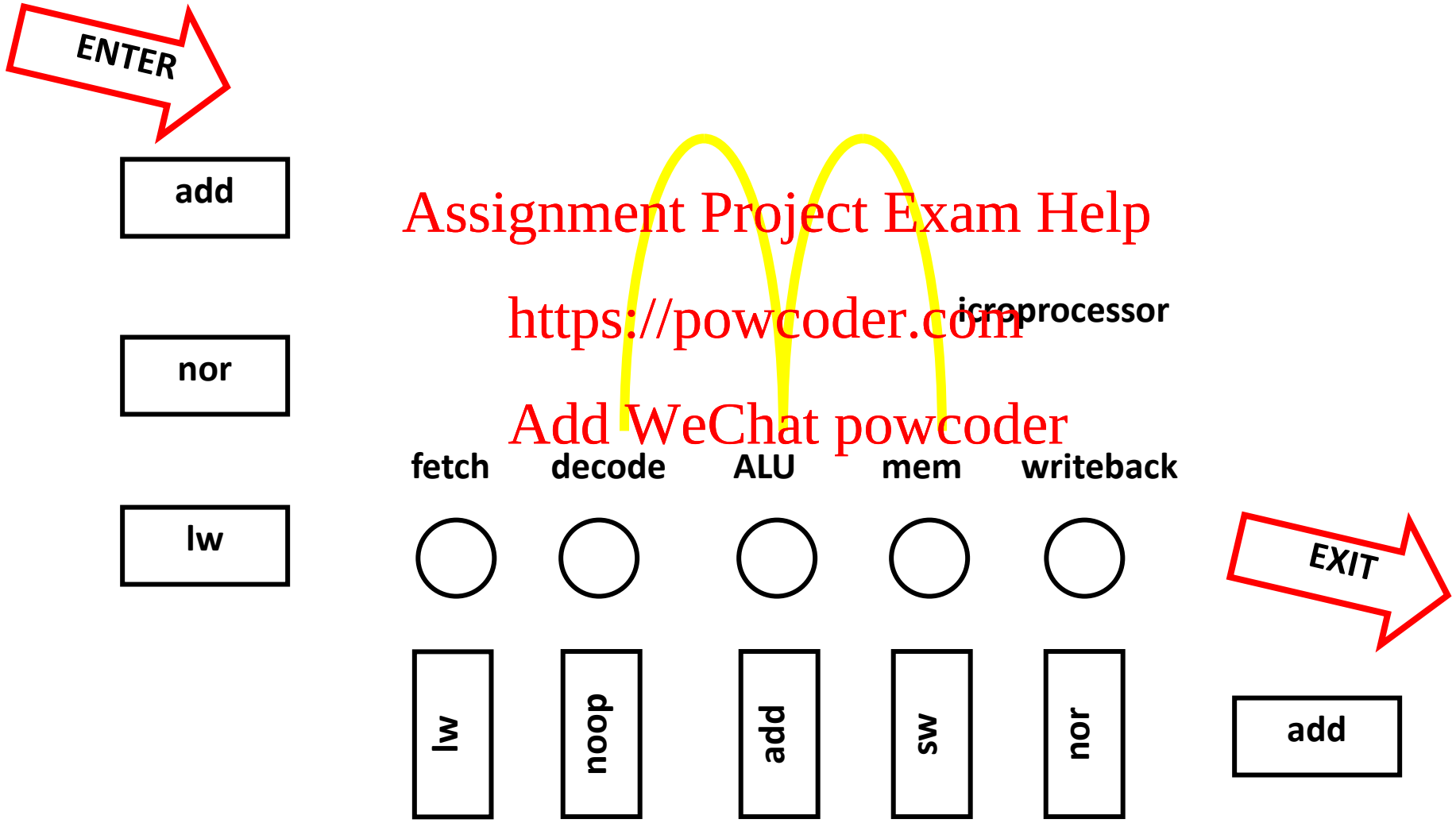
Ford magneto assembly line - 1913

Ford Wayne assembly line - 2011

Pipelining – Fast Food



Pipelining – Instruction Execution



Pipelining Today

- Execute as many instructions at the same time as possible.
  - Pipelining: 12-20+ cycles
  - Multiple pipelines
- Pentium:
  - 2 pipelines, 5 cycles each (10 instructions "in flight")
- Pentium Pro/II/III
  - 3 pipelines (kinda), 12 cycles each (kinda)
  - Instructions can execute out of their original program order
- Pentium IV
  - 4 pipelines, 20 cycles deep
  - Prescott: 4 pipelines, 31 cycles deep (could be clocked up to 8 GHz with special cooling)
- Core i7 (Nehalem)
  - 4 pipelines, 16 cycles deep

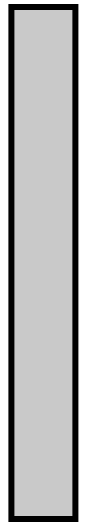
Pipelined implementation of LC2Kx

- Break the execution of the instruction into cycles.
  - Similar to the multi-cycle datapath
- Design a separate datapath stage for the execution performed during each cycle.
  - Build pipeline registers to communicate between the stages.

Aside: Pipeline Registers

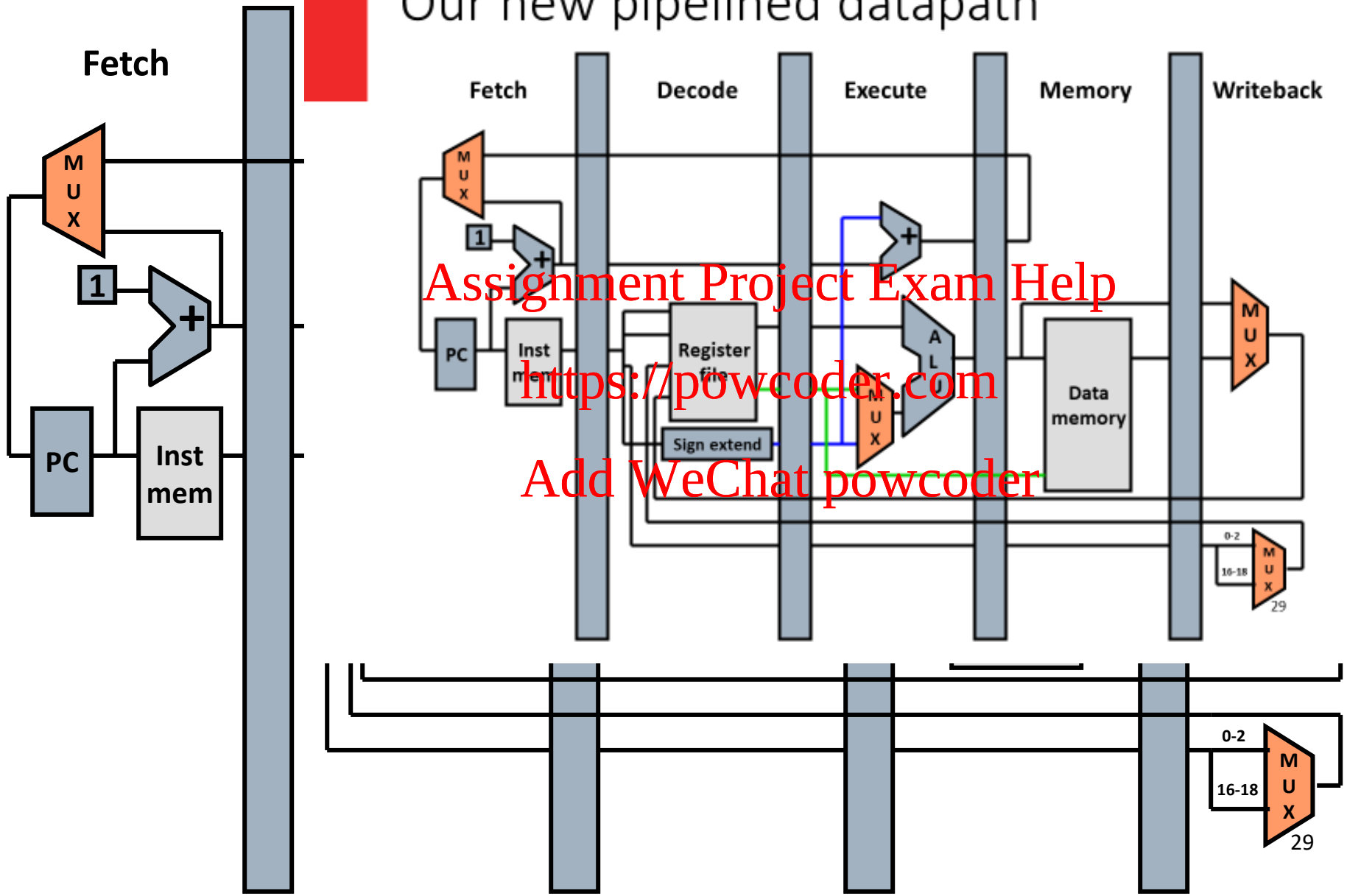
Since we're breaking operations into multiple cycles, we'll need extra **state** to keep track of data at each stage

- **Pipeline register**
  - It stores… whatever is relevant for that stage
  - Kind of like the **Instruction Register** in our multi-cycle design, but we will need one for each stage
  - Whatever goes in on the left gets saved and output on the right



Our new pipelined datapath

Pipelining

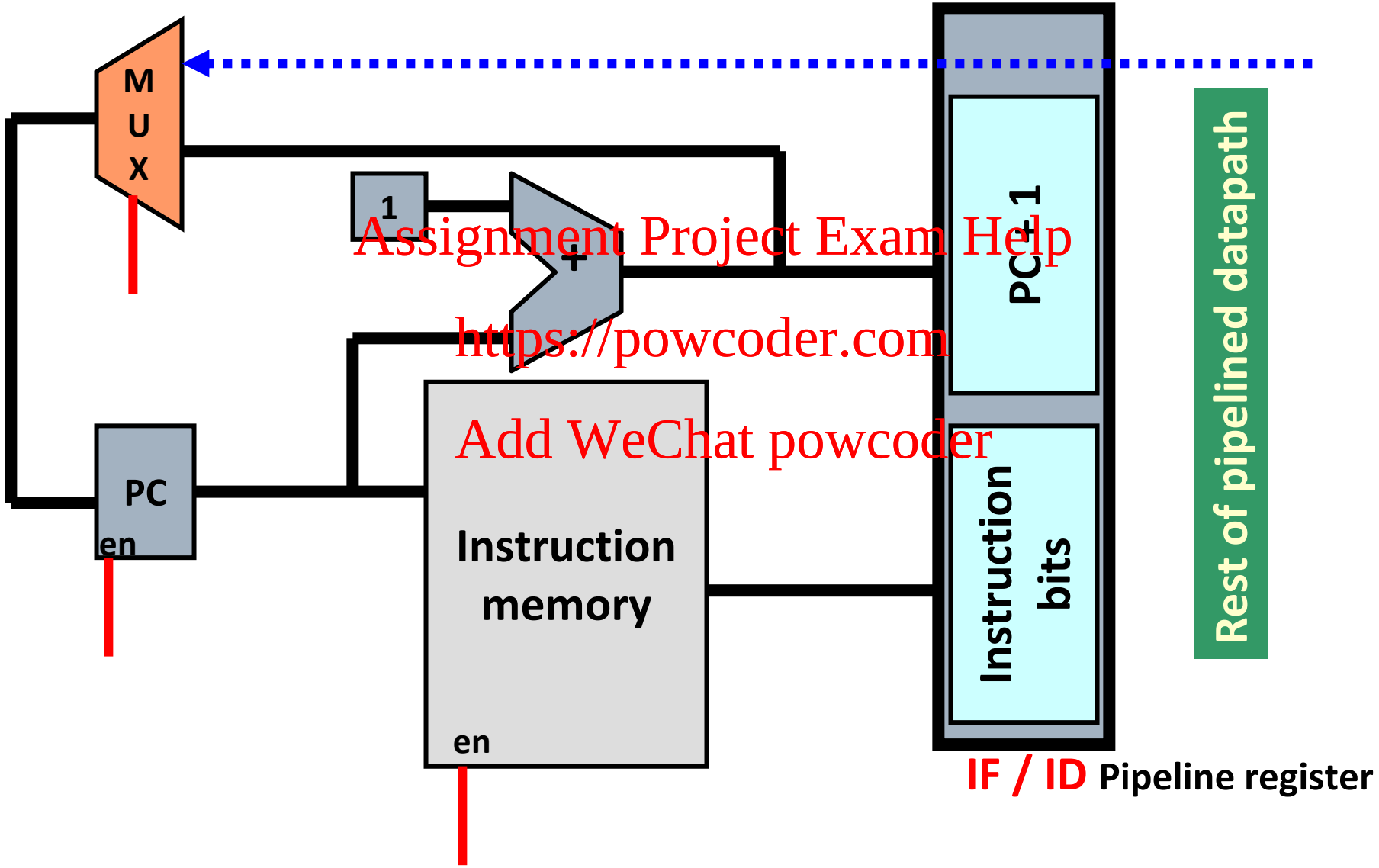


29

Stage 1: Fetch

- Design a datapath that can fetch an instruction from memory every cycle.
  - Use PC to index memory to read instruction
  - Increment the PC (assume no branches for now)
- Write everything needed to complete execution to the pipeline register (IF/ID)
  - The next stage will read this pipeline register.
  - Note that pipeline register must be edge-triggered

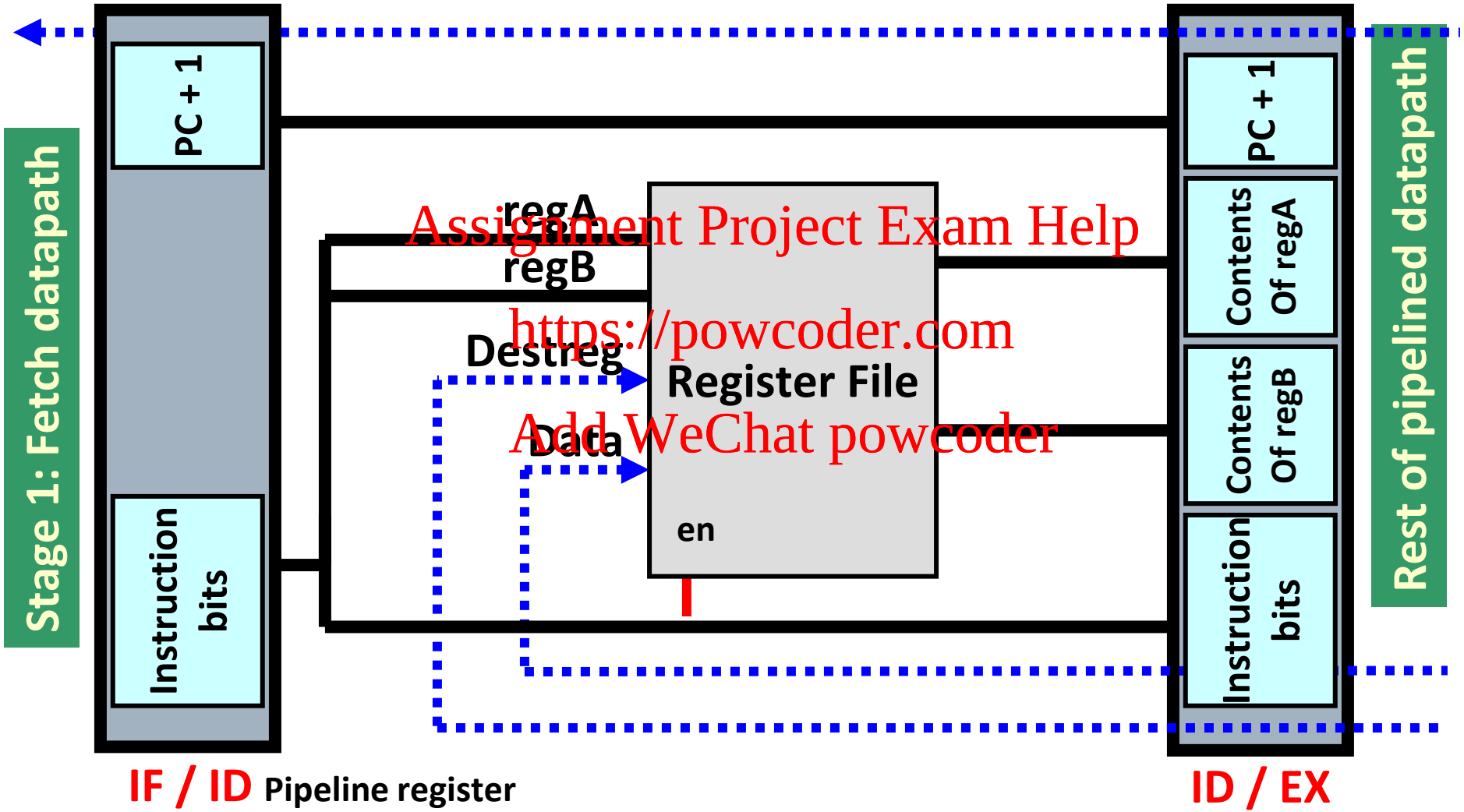
Pipeline Datapath – Fetch Stage



Stage 2: Decode

- Design a datapath that reads the IF/ID pipeline register, decodes instruction and reads register file (specified by regA and regB of instruction bits).
  - Decode is easy, just pass on the opcode and let later stages figure out their own control signals for the instruction.
- Write everything needed to complete execution to the pipeline register (ID/EX)
  - Pass on the offset field and both destination register specifiers (or simply pass on the whole instruction!).
  - Including PC+1 even though decode didn't use it.

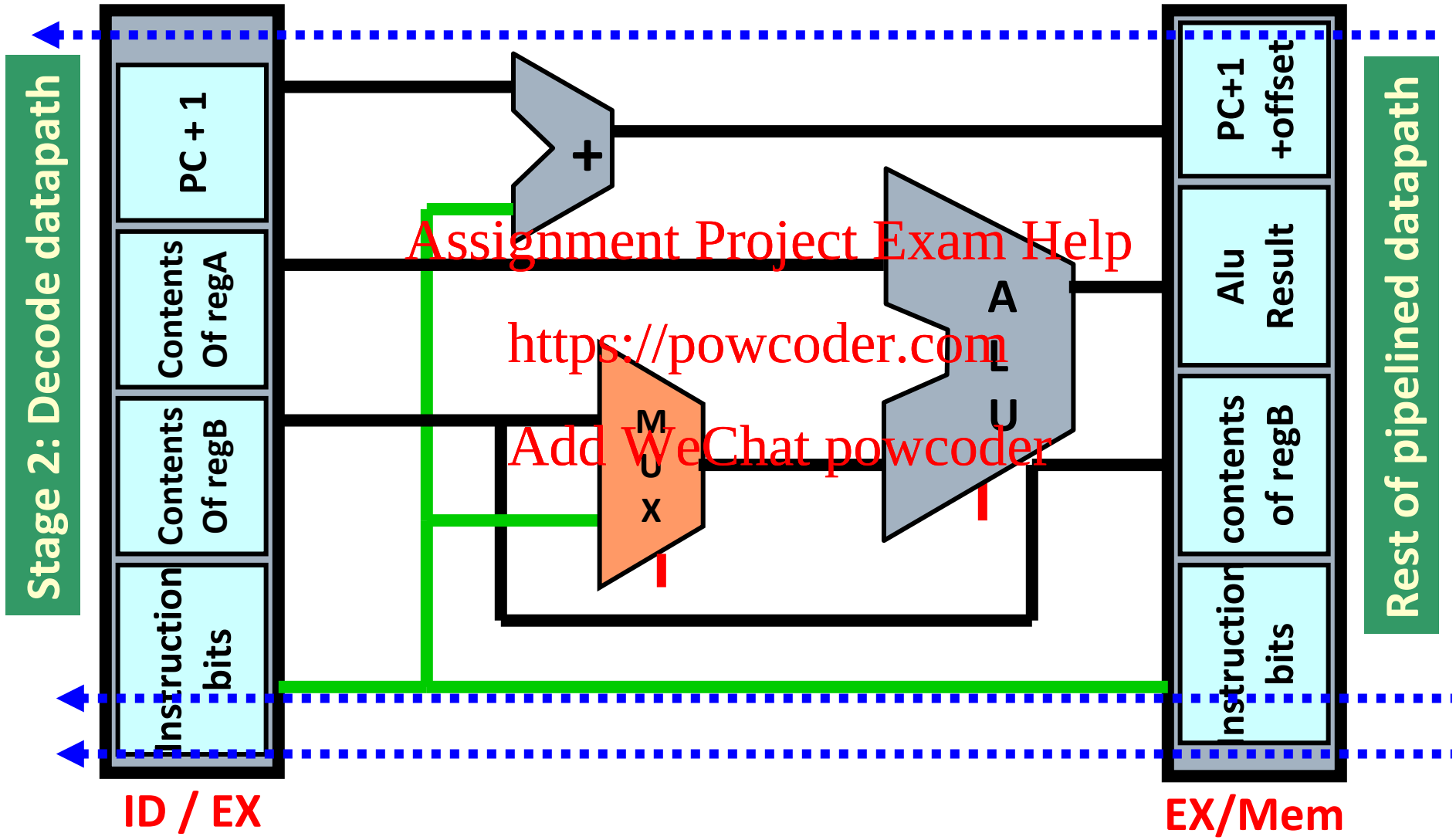
Pipeline Datapath – Decode Stage



Stage 3: Execute

- Design a datapath that performs the proper ALU operation for the instruction specified and the values present in the ID/EX pipeline register.
  - The inputs are the contents of regA and either the contents of regB or the offset field on the instruction.
  - Also, calculate PC+1+offset in case this is a branch.
- Write everything needed to complete execution to the pipeline register (EX/Mem)
  - ALU result, contents of regB and PC+1+offset
  - Instruction bits for opcode and destReg specifiers
  - Result from comparison of regA and regB contents

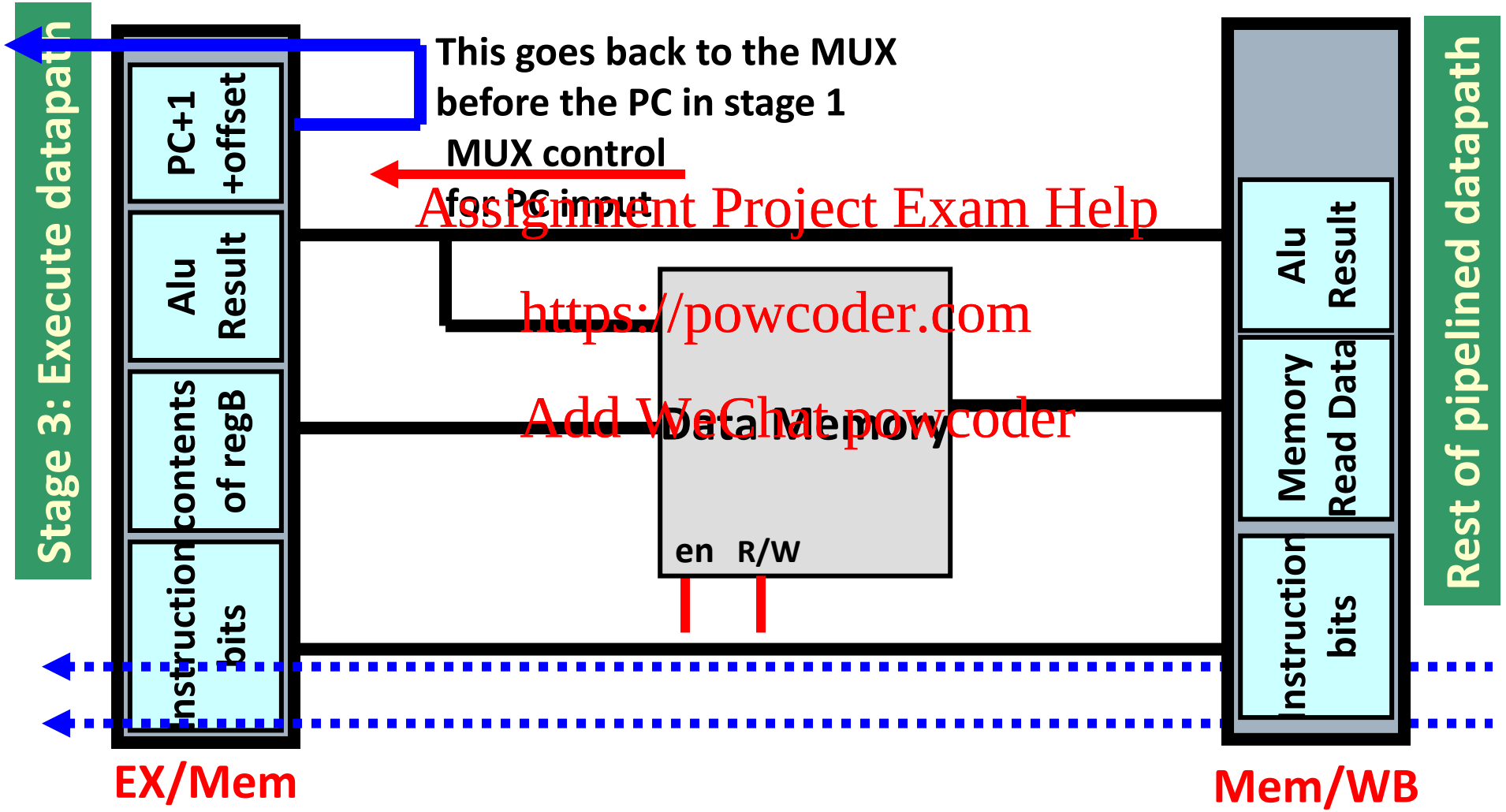
Pipeline Datapath – Execute Stage



Stage 4: Memory Operation

- Design a datapath that performs the proper memory operation for the instruction specified and the values present in the EX/Mem pipeline register.
  - ALU result contains address for **ld** and **st** instructions.
  - Opcode bits control memory R/W and enable signals.
- Write everything needed to complete execution to the pipeline register (Mem/WB)
  - ALU result and MemData
  - Instruction bits for opcode and destReg specifiers

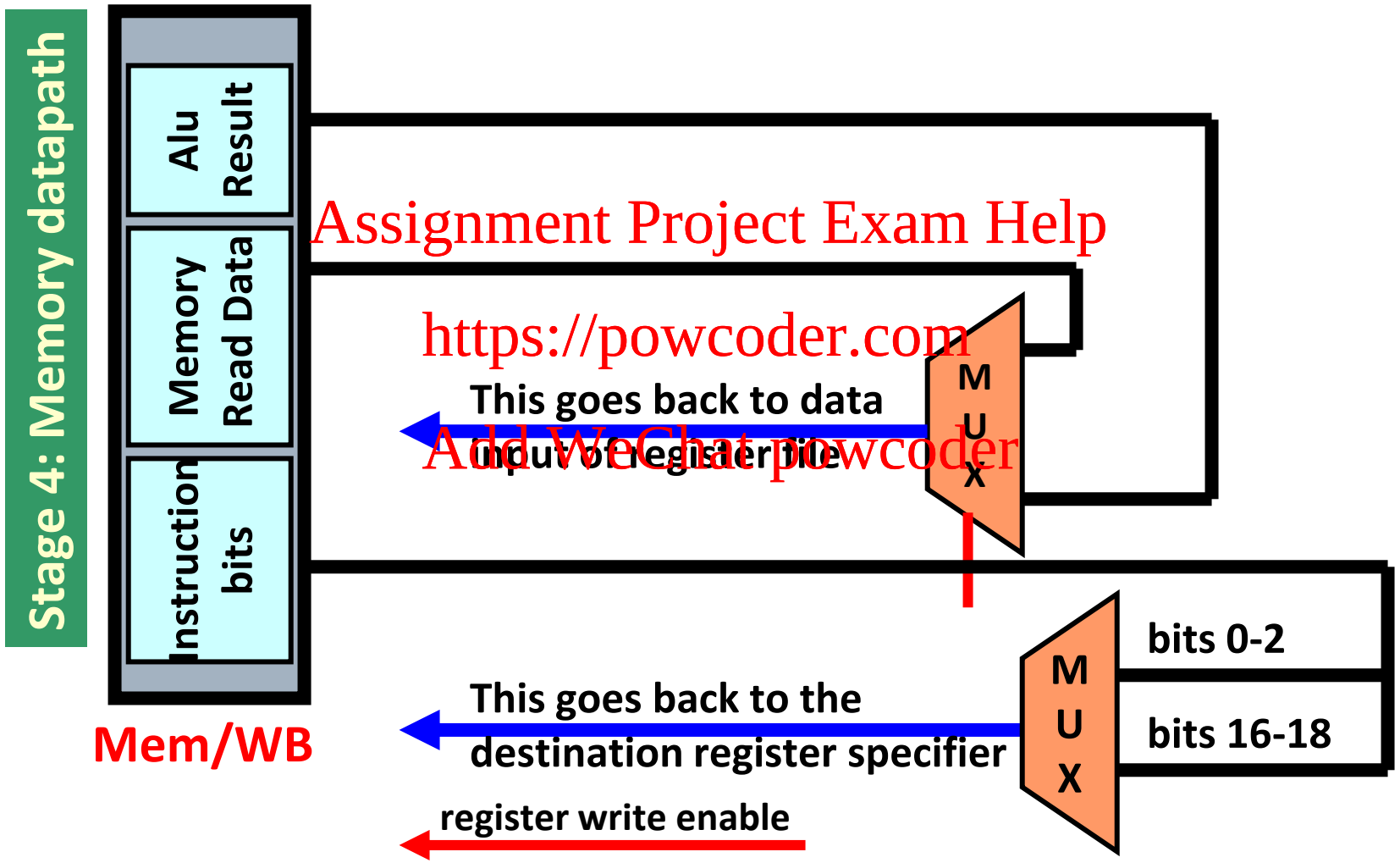
Pipeline Datapath – Memory Stage



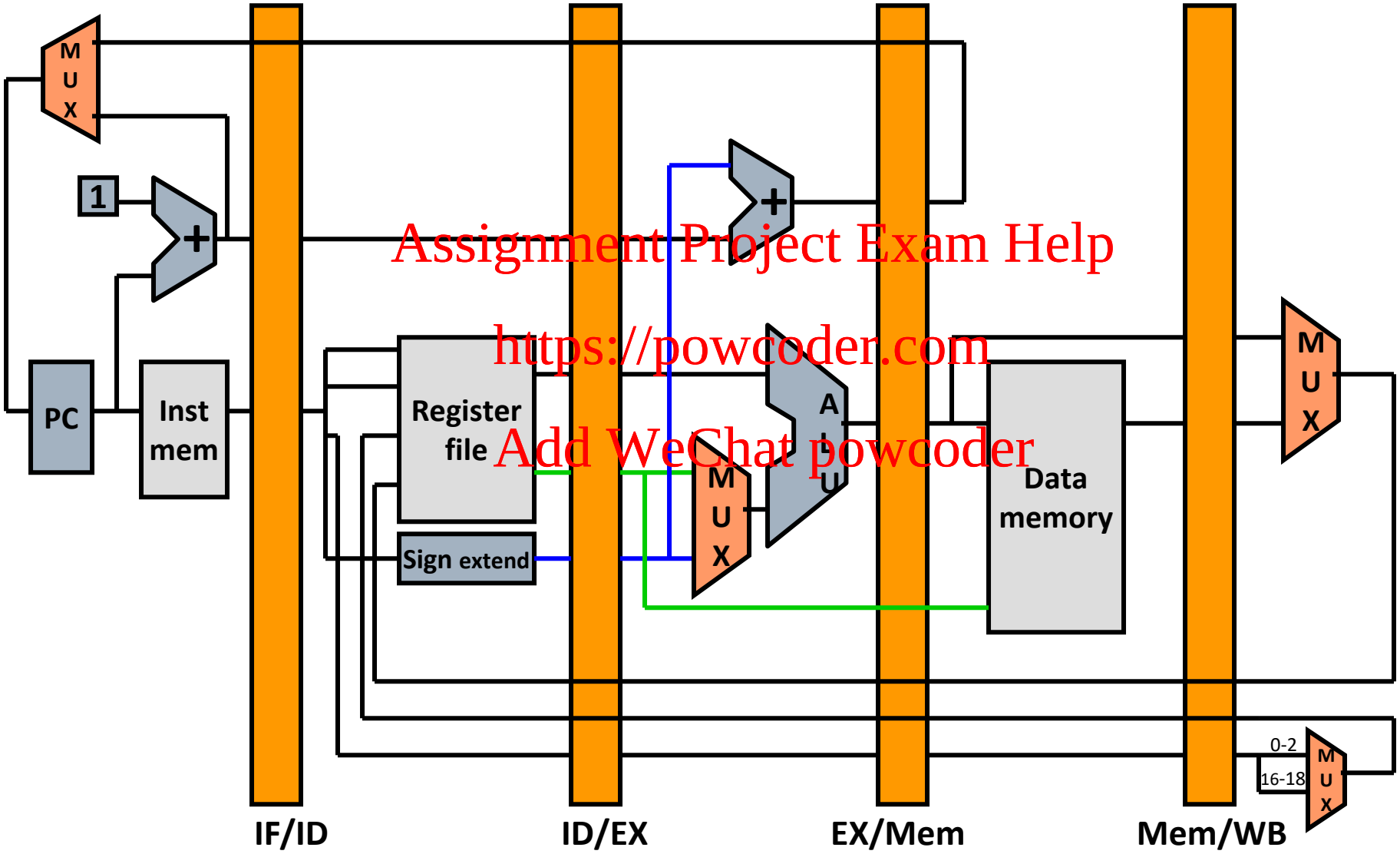
Stage 5: Write Back

- Design a datapath that completes the execution of this instruction, writing to the register file if required.
  - Write MemData to destReg for ld instruction.
  - Write ALU result to destReg for add or nor instructions.
  - Opcode bits also control register write enable signal.

Pipeline Datapath – Writeback Stage



Putting it All Together

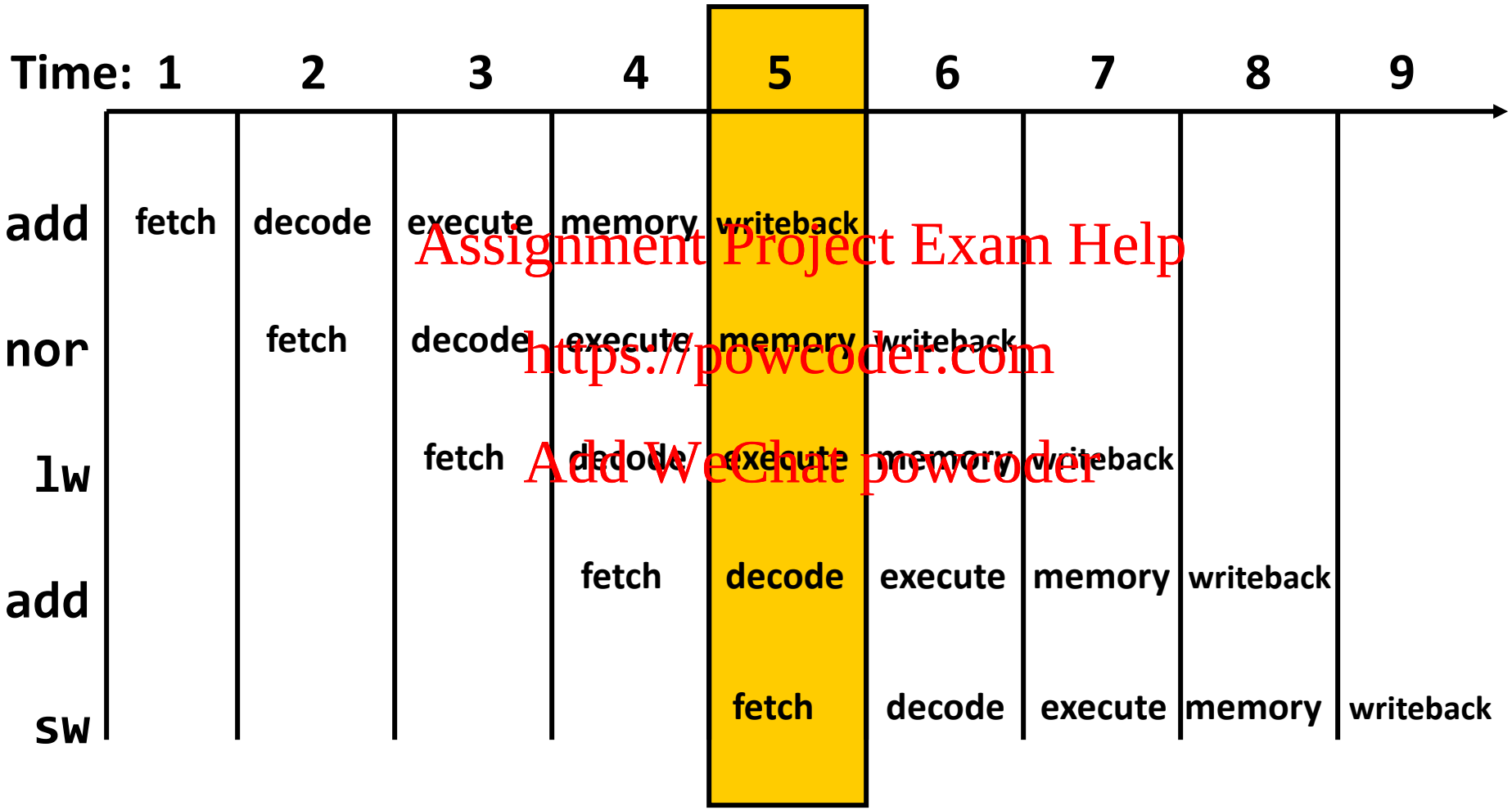


Sample Code (Simple)

Let us run the following code on pipelined LC2K2x:

```
add  1  2  3  ;  reg3 = reg1 + reg2
nor  4  5  6  ;  reg6 = reg4 nor reg5
lw   2  4  20 ;  reg4 = Mem[reg2 + 20]
add  2  5  5  ;  reg5 = reg2 + reg5
sw   3  7  10 ;  Mem[reg3 + 10] = reg7
```

Time Graphs (a.k.a. Pipe Trace)



A vertical slice reports the entire activity of the pipeline at time 5

Logistics

- There are 3 videos for lecture 12
  - L12_1 – Performance_Metrics
  - L12_2 – Pipelining_Introduction
  - L12_3 – Pipelining_Execution-Example (really 12_2 part 2)
- There is one worksheet for lecture 12
  1. L12 worksheet