RISC-V Pipelined Architecture I

Muhammad Tahir

Lecture 9

Electrical Engineering Department
University of Engineering and Technology Lahore

Contents

Instruction Types and Usage of Functional Units

Table 1: Functional units used by different instruction types.

Instruction	IF Unit	ID Unit	ALU	MEM Access	WB
R-type	Yes	Yes	Yes		Yes
I-type	Yes	Yes	Yes		Yes
I-type (Loads)	Yes	Yes	Yes	Yes	Yes
S-type	Yes	Yes	Yes	Yes	
B-type	Yes	Yes	Yes		
U-type	Yes	Yes	Yes		Yes
J-type	Yes	Yes	Yes		Yes

- For single cycle implementation, worst case cycle time is for Load instruction
- None of the other instructions involves all 5 functional units

Multi-cycle Implementation

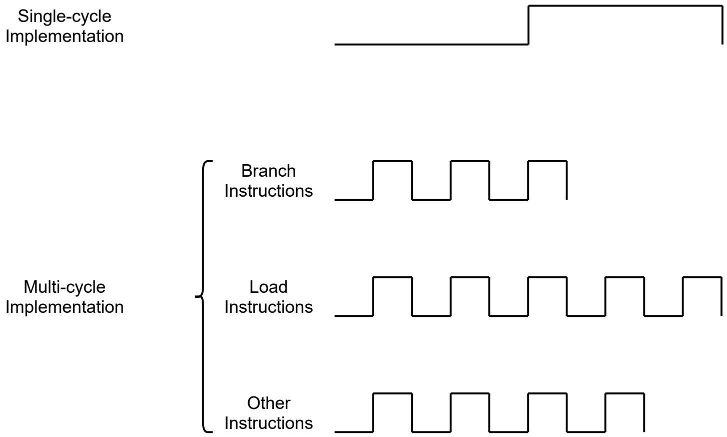
- Single Cycle Processor
  - The cycle time has to be long enough for the slowest instruction to complete
  - CPI $= 1$, but cycle time is very long
- Multi-cycle Implementation
  - Break instructions into multiple smaller steps
  - Each step executes in one cycle (but smaller cycle time)
  - The smaller cycle time has to be long enough to manage the slowest step
  - Another advantage is reuse of some (hardware) resource(s)

Multi-cycle Implementation Cont'd

- In a multi-cycle processor
  - Cycle time is much shorter
  - Different instructions take different number of cycles to complete
- CPI for multi-cycle implementation
  - Load instruction requires five cycles to complete
  - But branch takes only three cycles
  - Other instructions take four cycles
  - CPI falls between 3 to 5 for multi-cycle implementation

Multi-cycle Implementation Cont'd

- How multi-cycle compare against single cycle



Multicycle Implementation Cont'd

- Recall

$$\frac{time}{program} = \frac{instructions}{program} x \frac{cycles}{instruction} x \frac{time}{cycle}$$

OR

$$\text{Execution Time} = N \times CPI \times \frac{time}{cycle}$$

- For multi-cycle, although $CPI \uparrow$ but $\frac{time}{cycle} \downarrow$
- Overall execution time is shortened and performance is improved
- Performance can further be improved with pipelining

Pipelined Implementation

- Pipelining does not reduce time-to-completion for a single operation (instruction), but it increases the throughput of the processor
- Multiple operations (different) are performed simultaneously using different resources
    - Time-to-completion: Time required to complete a certain operation
    - Throughput: Amount of work that can be performed over a period of time

Why Pipelining is Needed: Assembly Line Example

- A car assembly line is an example of pipelining
- Assume there are four phases to assemble a car and each phase takes *5 minutes* to complete
    - Paint the body
    - Put in an engine
    - Put on doors and hood
    - Put on wheels

Assembly Line Example Cont'd

- Suppose there is only one person working on the assembly line
- The person will do the work at every station by moving along with the car to each station
- It would take 20 minutes to complete one car

Table 2: Cars Assembly

Time	Engine	Doors	Wheels	Paint
5 min	Car1			
10 min		Car1		
15 min			Car1	
20 min				Car1

Assembly Line Example Cont'd

- It takes double time to complete two cars

Table 3: Cars Assembly

Time	Engine	Doors	Wheels	Paint
5 min	Car1			
10 min		Car1		
15 min			Car1	
20 min				Car1
25 min	Car2			
30 min		Car2		
35 min			Car2	
40 min				Car2

Pipelining in Assembly Line

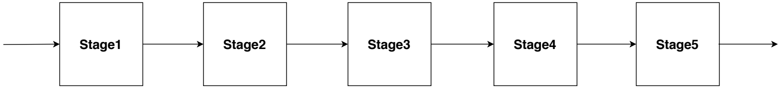
- Pipelined assembly line

Table 4: Cars Assembly

Time	Engine	Doors	Wheels	Paint
5 min	Car1			
10 min	Car2	Car1		
15 min	Car3	Car2	Car1	
20 min	Car4	Car3	Car2	Car1
25 min		Car4	Car3	Car2
30 min			Car4	Car3
35 min				Car4

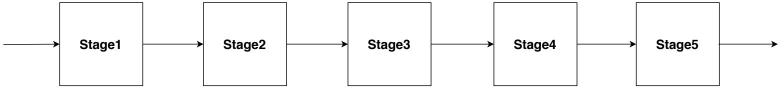
Pipelined Implementation Cont'd

- How to move from stage to stage?

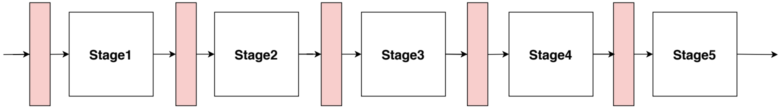


Pipelined Implementation Cont'd

- How to move from stage to stage?

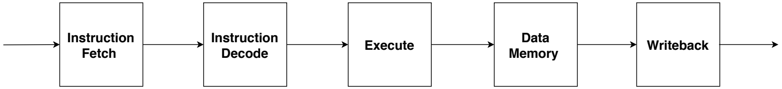


- Move from stage to stage using Buffers/Registers

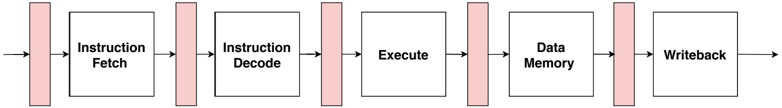


Five Stage RISC-V Pipeline

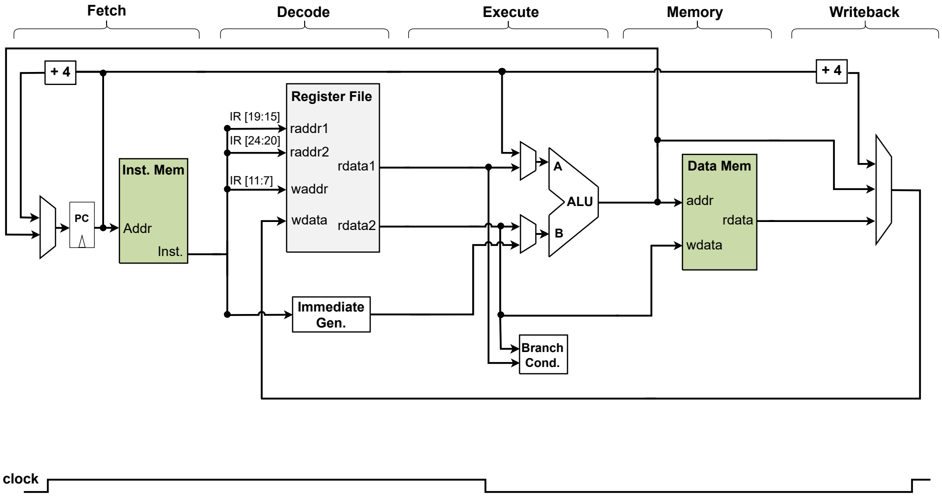
- RISC-V stages for instruction execution.



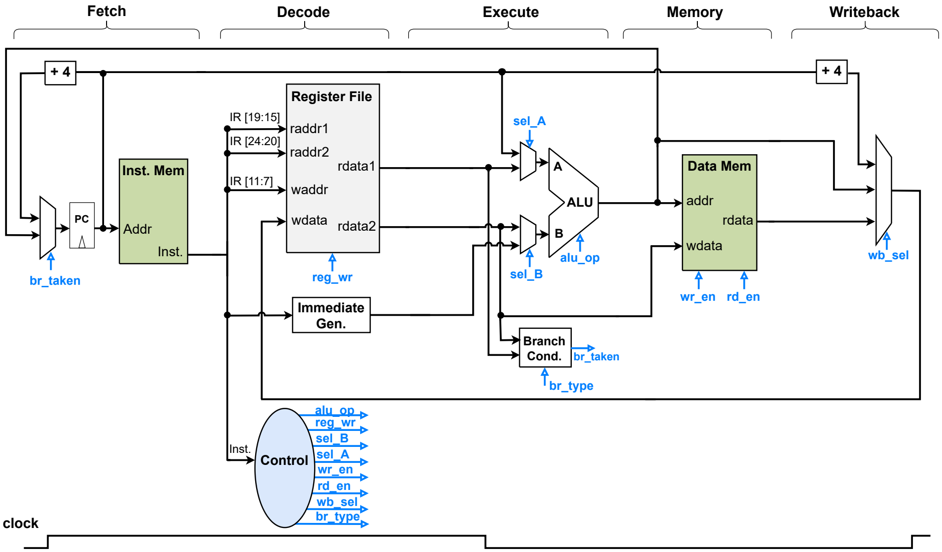
- Five stage RISC-V pipeline for instruction execution.



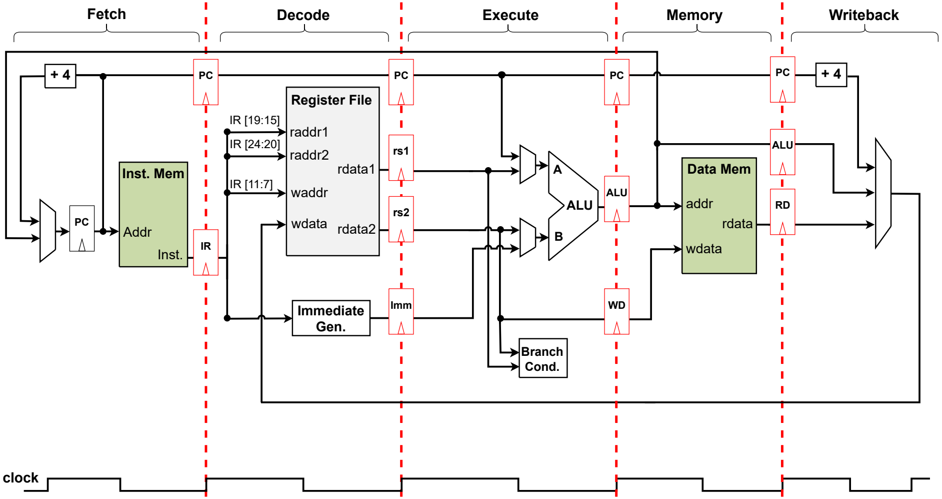
Single Cycle Datapath for RV32I



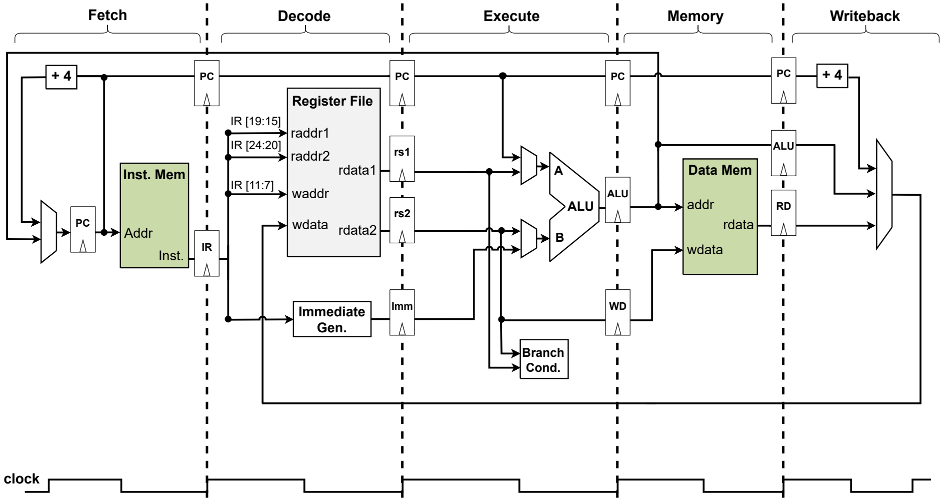
Single Cycle Datapath and Controller for RV32I



From Single Cycle to Pipelined Architecture



From Single Cycle to Pipelined Architecture Cont’d

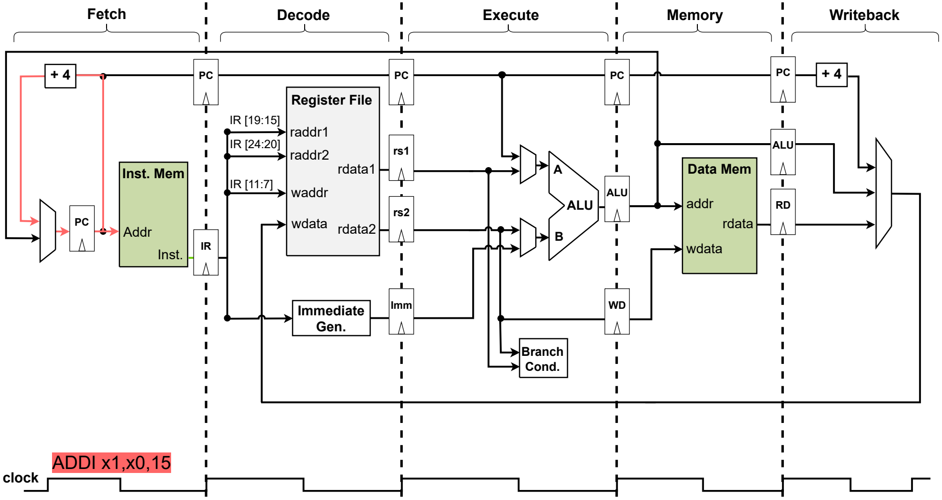


Assembly Instructions for Pipelined Execution

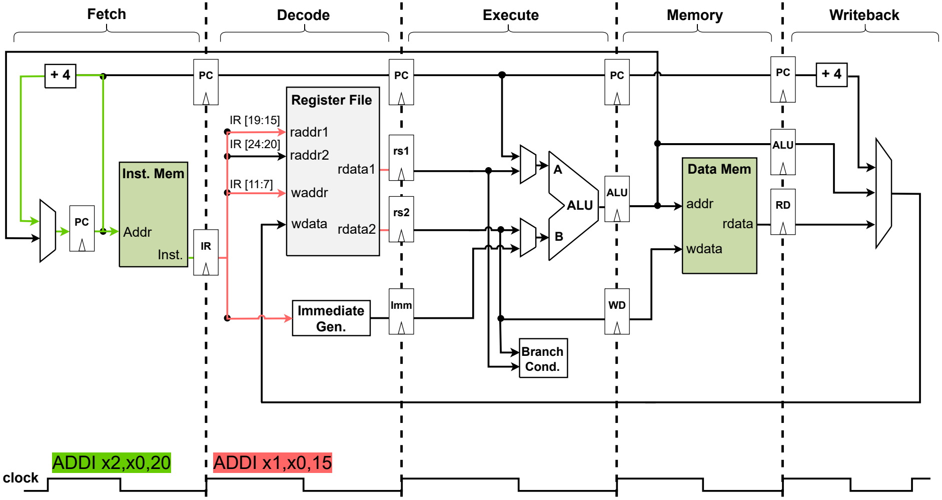
Example program.

```
addi  x1,  x0,  15
addi  x2,  x0,  20
add   x3,  x5,  x4
or    x4,  x6,  x8
and   x5,  x1,  x6
add   x8,  x7,  x2
```

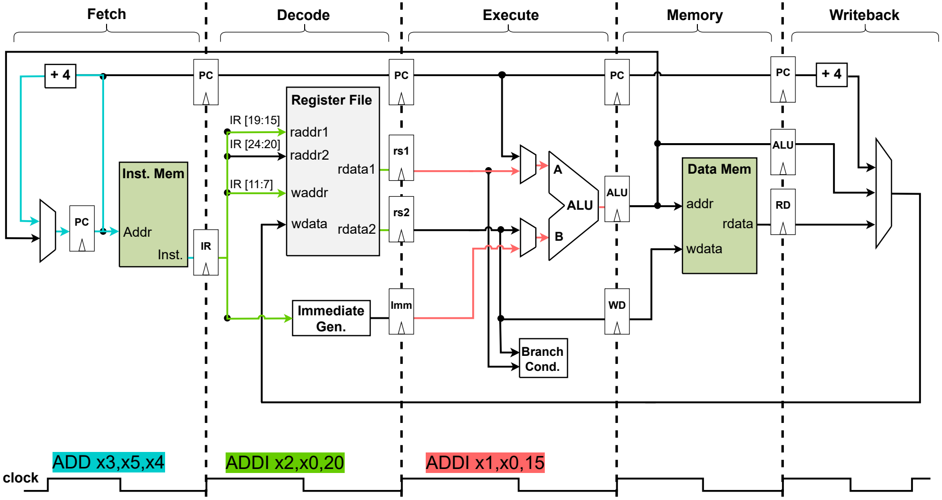
Pipelined Execution



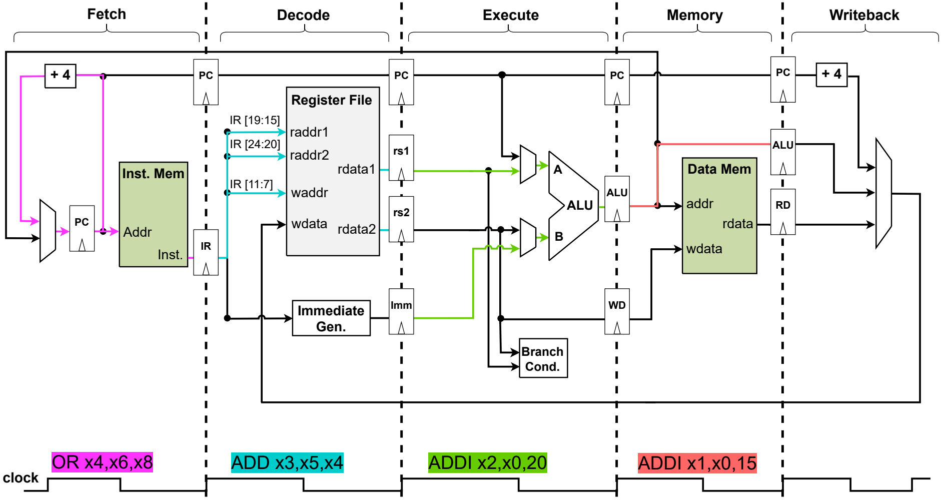
Pipelined Execution Cont'd



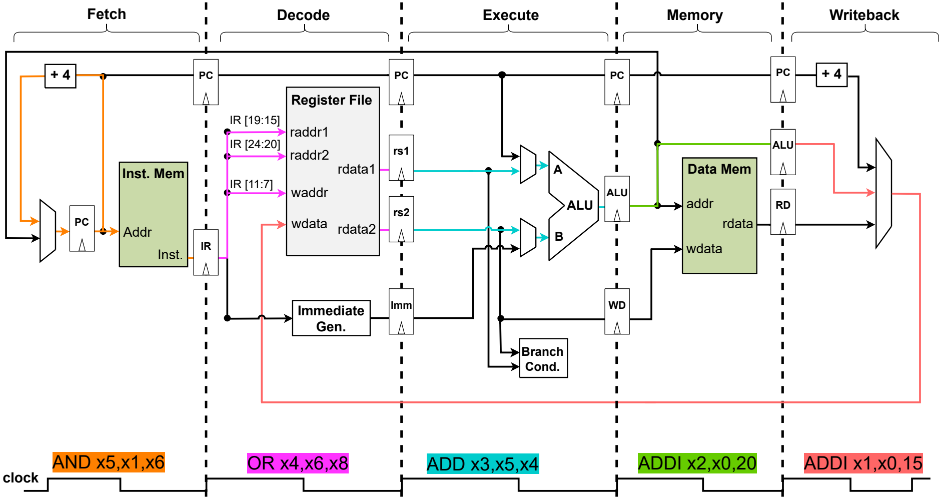
Pipelined Execution Cont'd



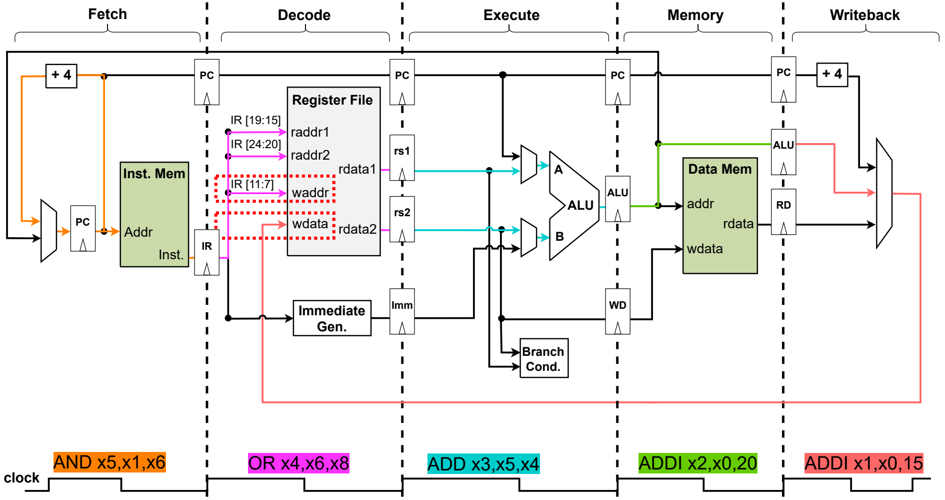
Pipelined Execution Cont'd



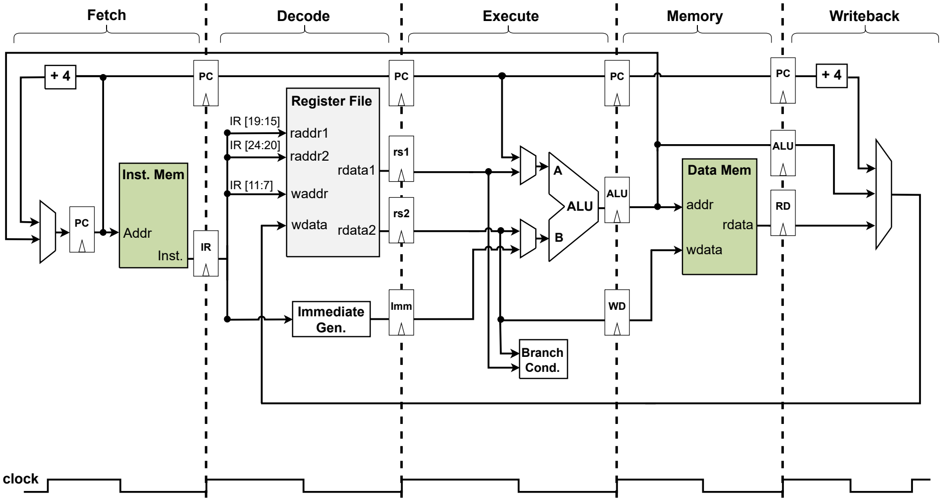
Pipelined Execution Cont’d



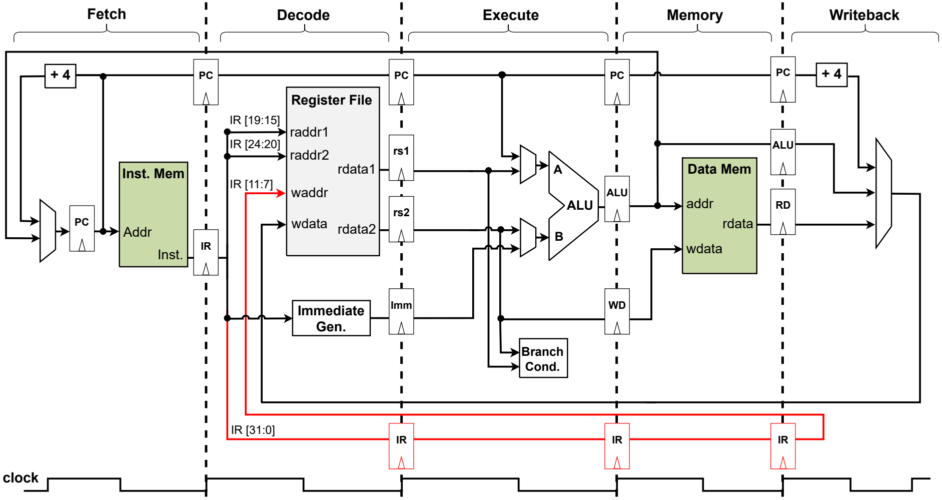
Pipelined Execution Cont'd



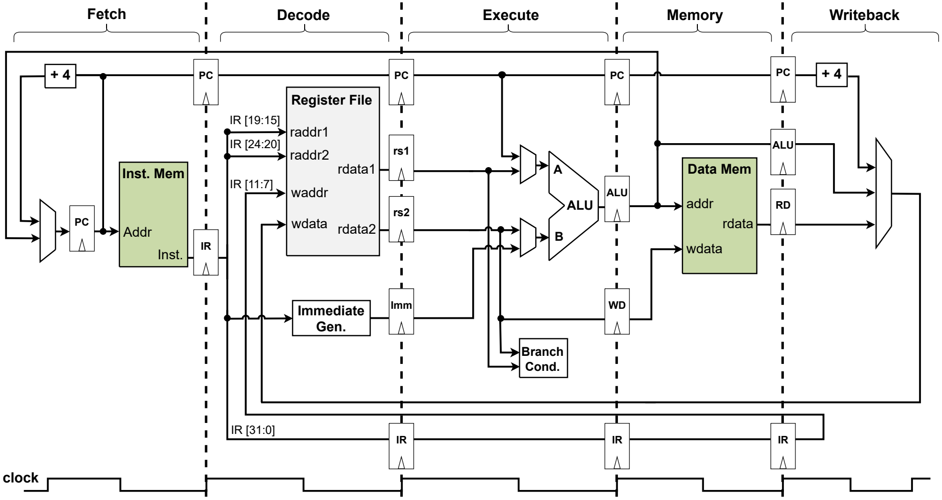
Pipelined Execution: Need for Modification



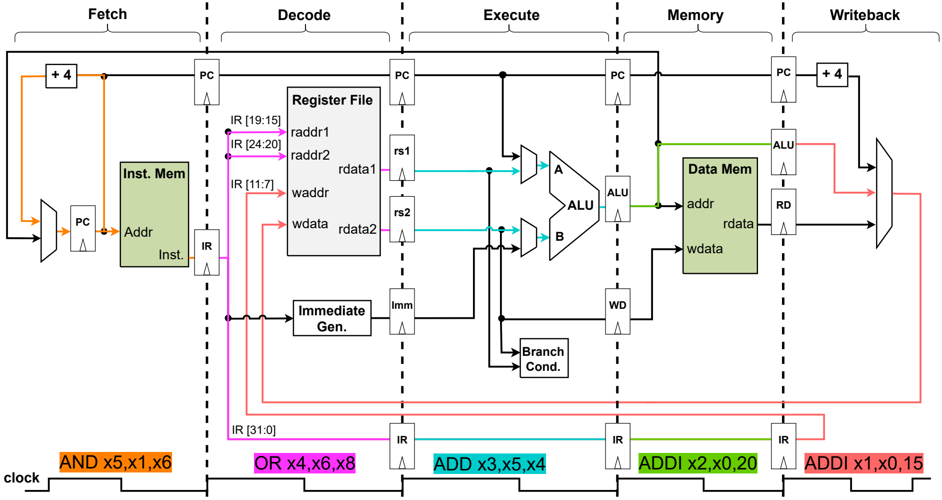
Pipelined Execution: The Modification



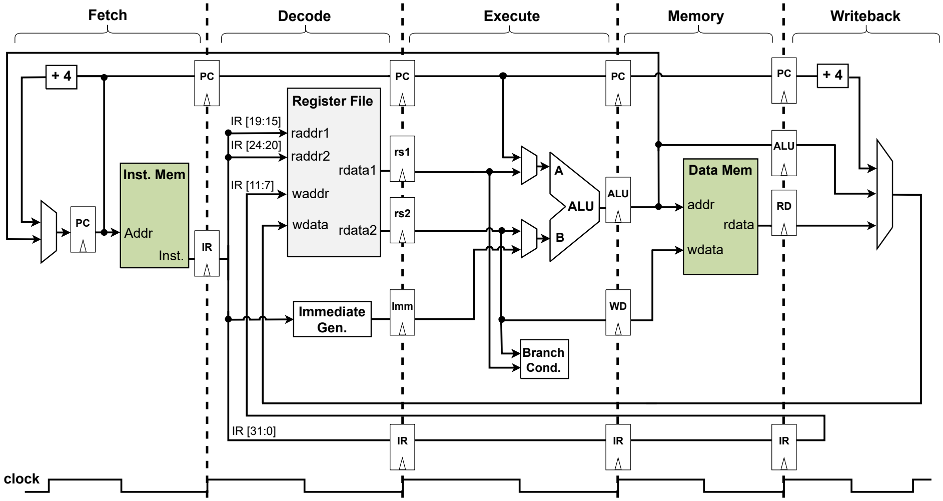
Pipelined Execution: Updated Datapath



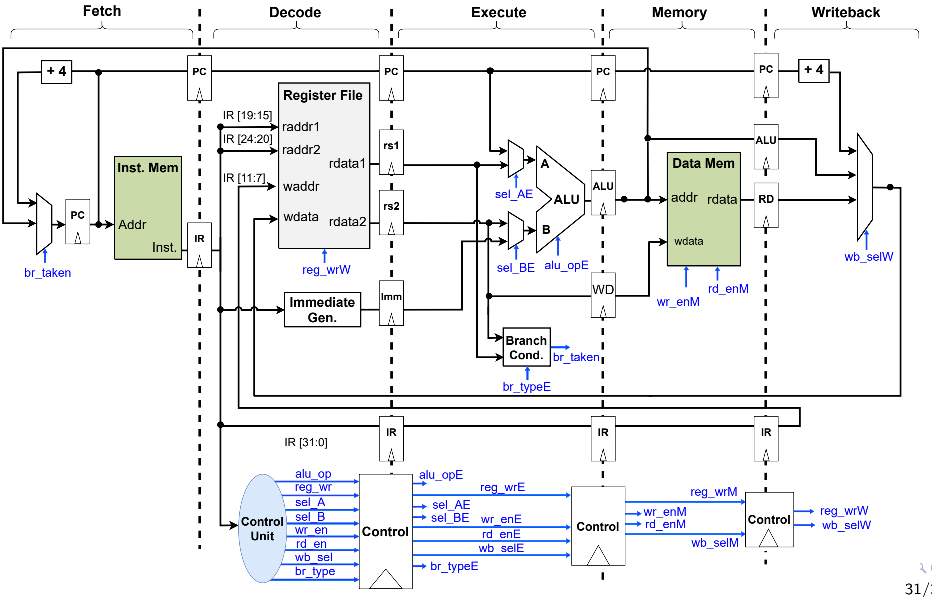
Modified Pipelined Execution



Complete Pipelined Datapath



Complete Pipelined Datapath with Control



Suggested Reading

- Read relevant sections of Chapter 4 of [Patterson and Hennessy, 2021].

Acknowledgment

- Preparation of this material was partly supported by Lampro Mellon Pakistan.

References

Patterson, D. and Hennessy, J. (2021).
Computer Organization and Design RISC-V Edition: The Hardware Software Interface, 2nd Edition.
Morgan Kaufmann.