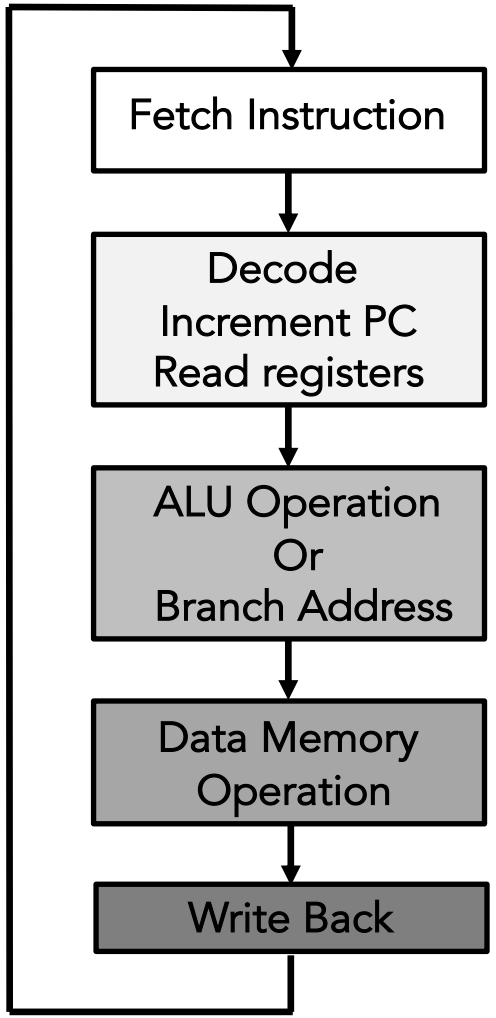
CSE 520
Computer Architecture II

Processor Pipelining

Prof. Michel A. Kinsy

Central Processing Unit (CPU)

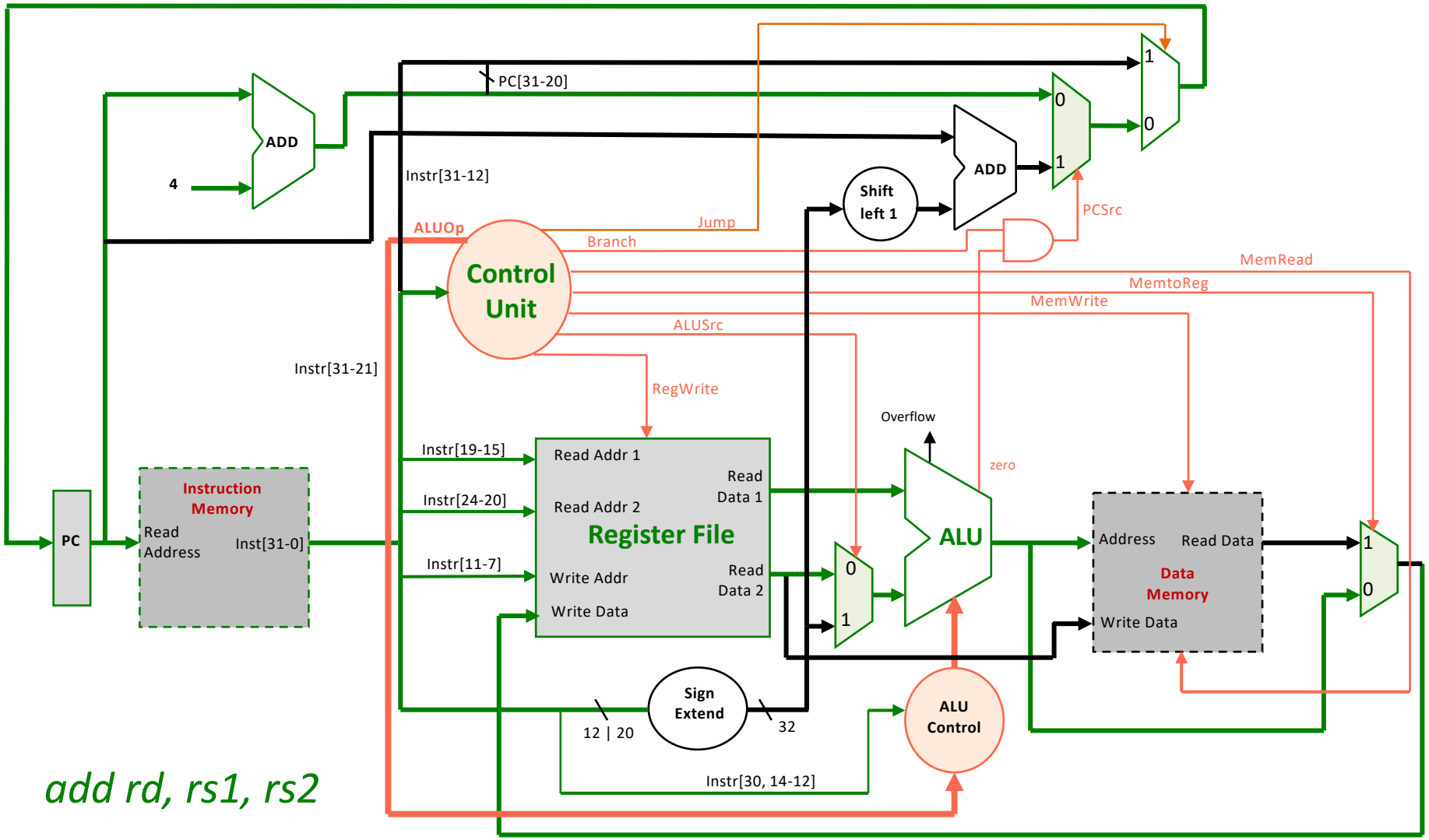
- Central Processing Unit (CPU) Organization
- CPU Execution Process
  1. Fetch Instruction
  2. Decode Instruction
  3. Execute Operation
  4. Memory Operation
  5. Register Writeback Operation



CPU Control

Signal name	Effect when de-asserted	Effect when asserted
RegDst	The destination register number comes from ***rs2***	The destination register number comes from ***rd***
RegWrite	None	Destination register is written with value on ***Write data***
ALUSrc	2nd ALU operand comes from ***Read Data 2***	2nd ALU operand is the sign extended, of some bits of the instruction
PCSrc	The ***PC*** is replaced by ***PC + 4***	The ***PC*** is replaced by the branch target address
MemRead	None	Memory is read
MemWrite	None	Memory is written
MemtoReg	The value to the register ***Write data*** input comes from the ALU	The value to the register ***Write data*** input comes from the data memory

CPU: R-Type Execution Datapath

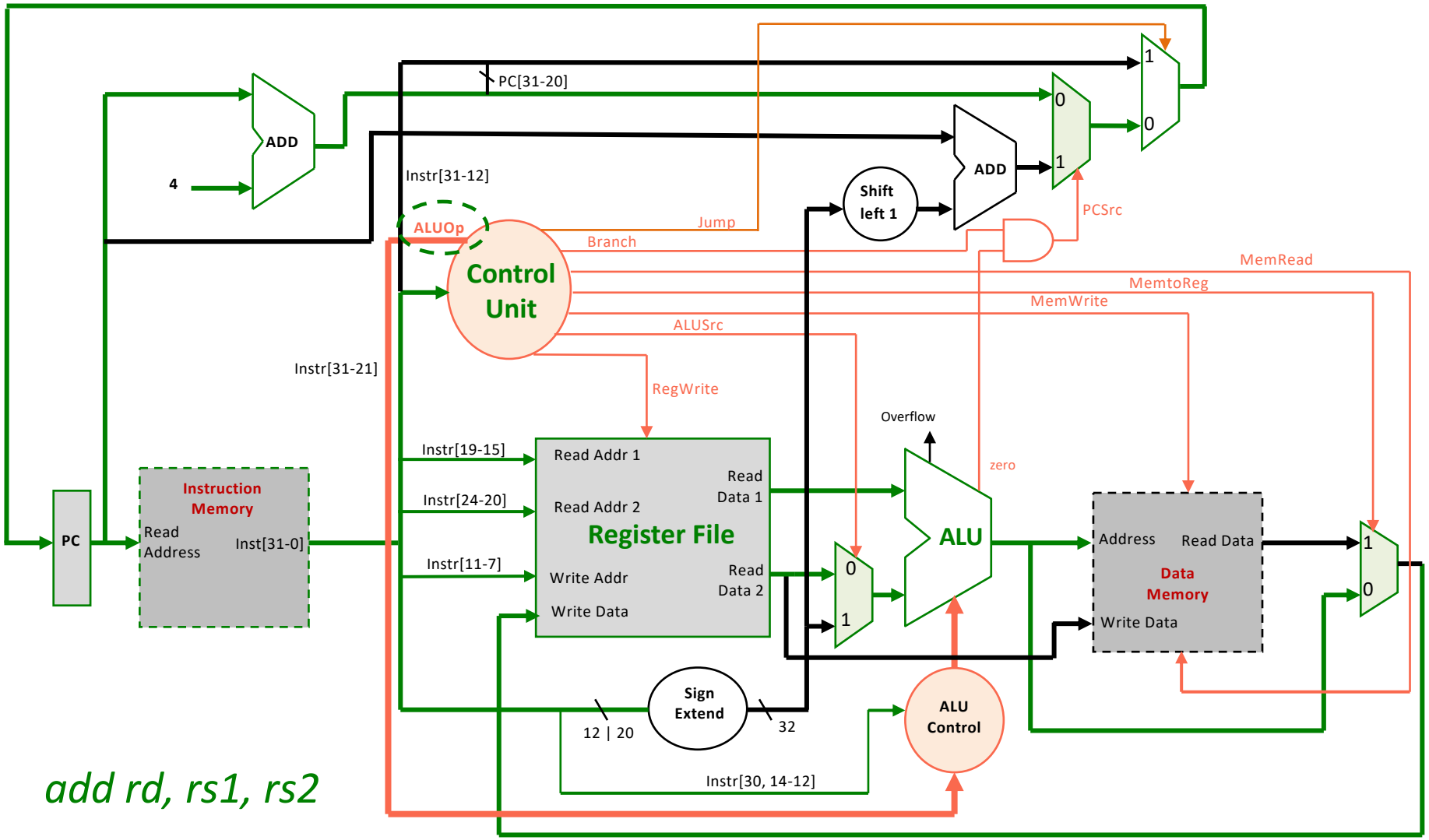


add rd, rs1, rs2

Instruction Execution Flows

Instruction class	Functional Units used by the instruction class				
R-type	Instruction fetch	Register access	ALU	Register access	

CPU: R-Type Execution Datapath

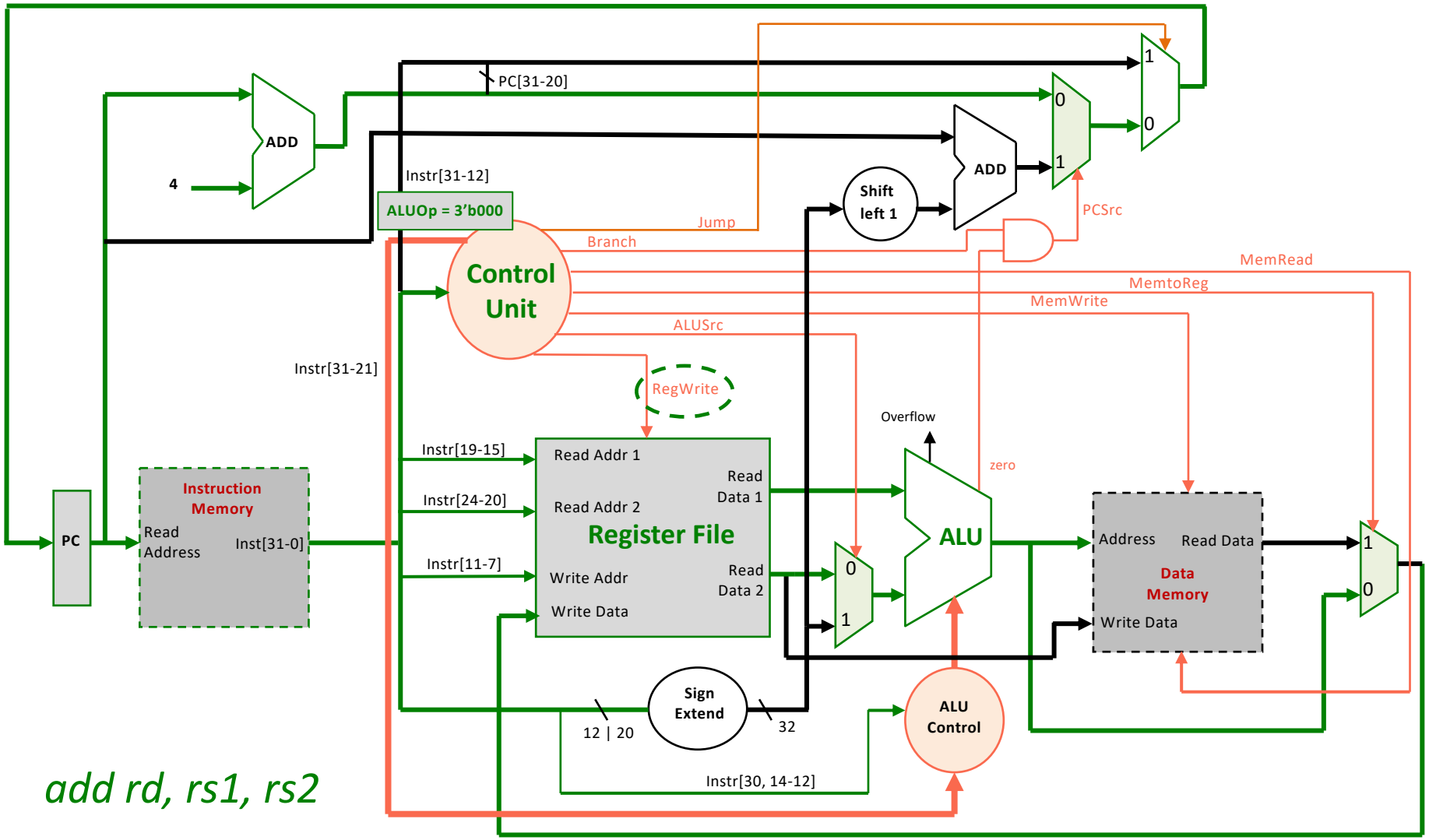


add rd, rs1, rs2

ALU Operation

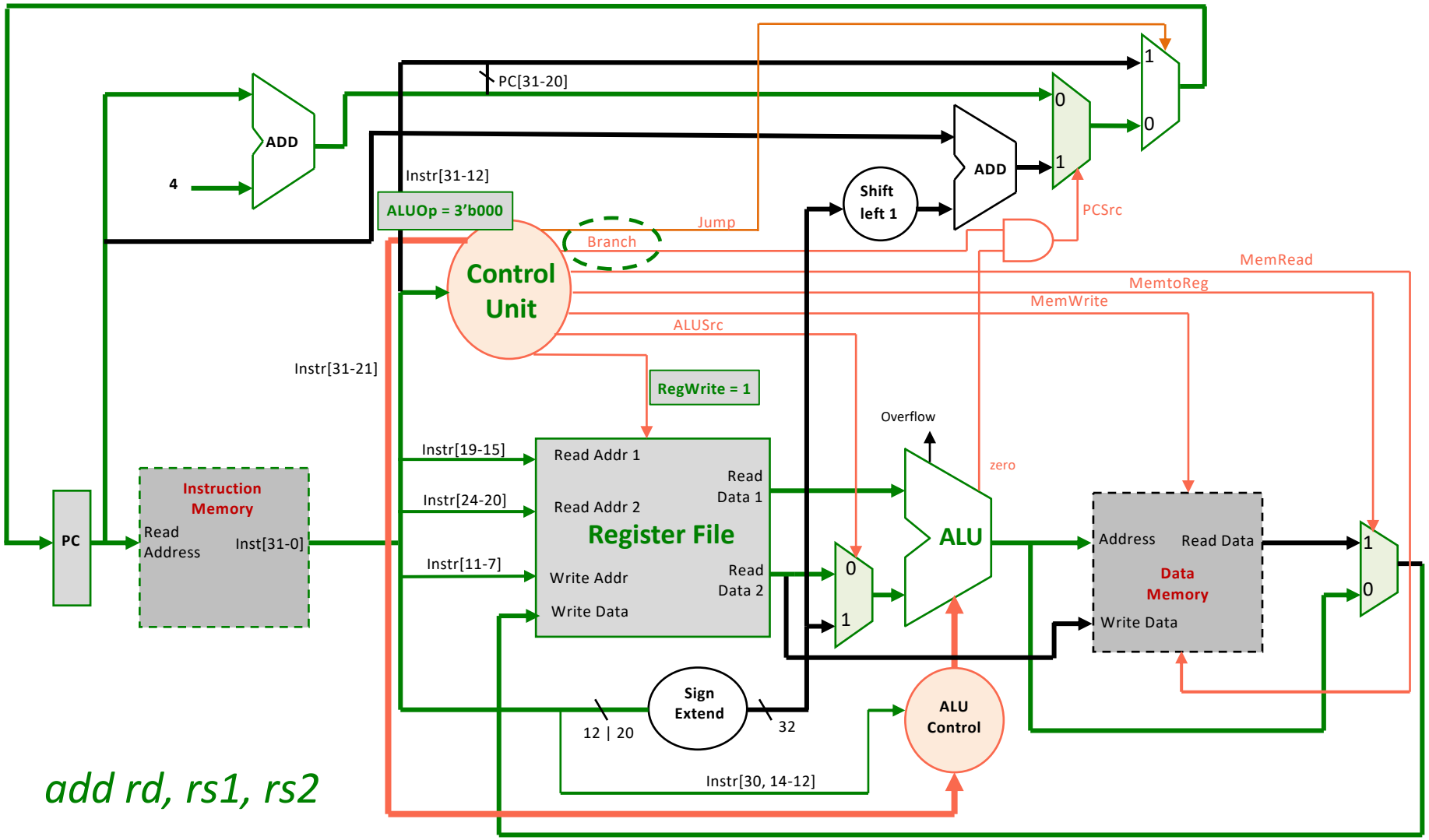
Instruction Opcode	Instruction Operation	ALU Op	funct3	funct7	Effective ALU Operation	ALU Control
LW	Load word	100	010	xxxxxxx	Add	000000
SW	Store word	101	010	xxxxxxx	Add	000000
BEQ	Branch equal	010	000	xxxxxxx	Subtract	001000
R-type	ADD	000	000	0000000	Add	000000
R-type	SUB	000	000	0100000	Subtract	001000
R-type	AND	000	111	0000000	and	000111
R-type	OR	000	110	0000000	Or	000110
R-type	SLT	000	010	0000000	Set-less-than	000010

CPU: R-Type Execution Datapath



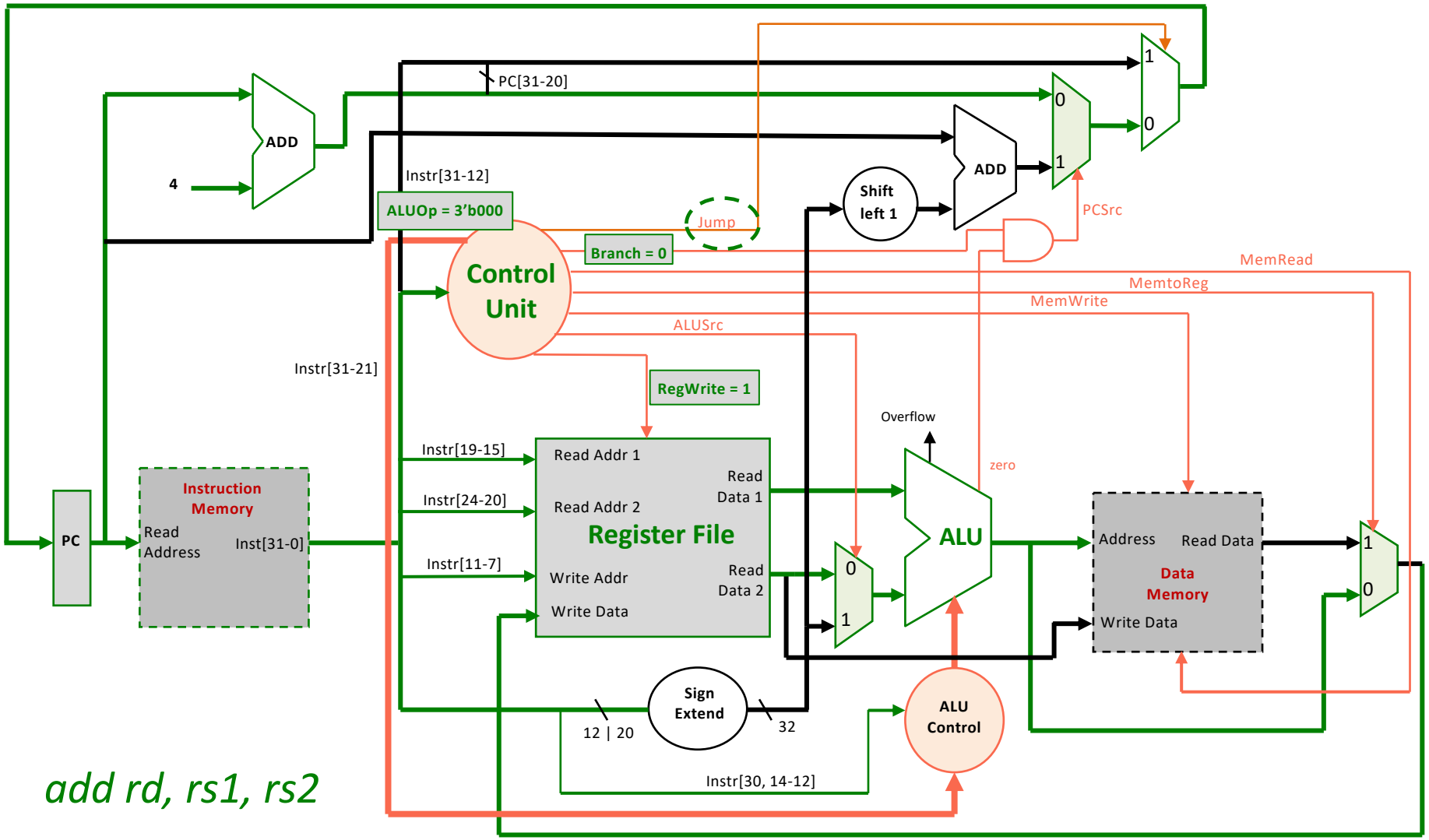
add rd, rs1, rs2

CPU: R-Type Execution Datapath



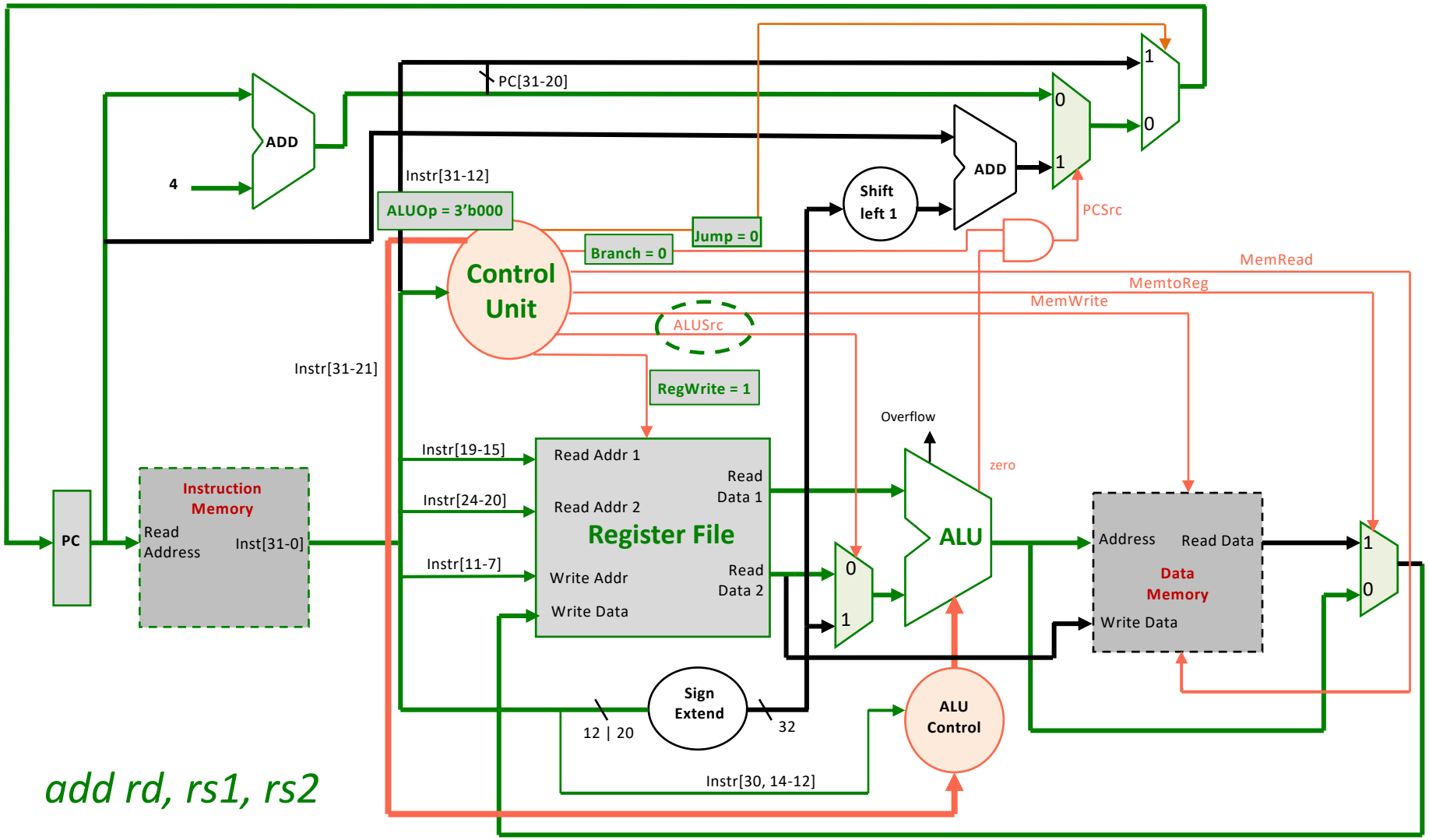
add rd, rs1, rs2

CPU: R-Type Execution Datapath



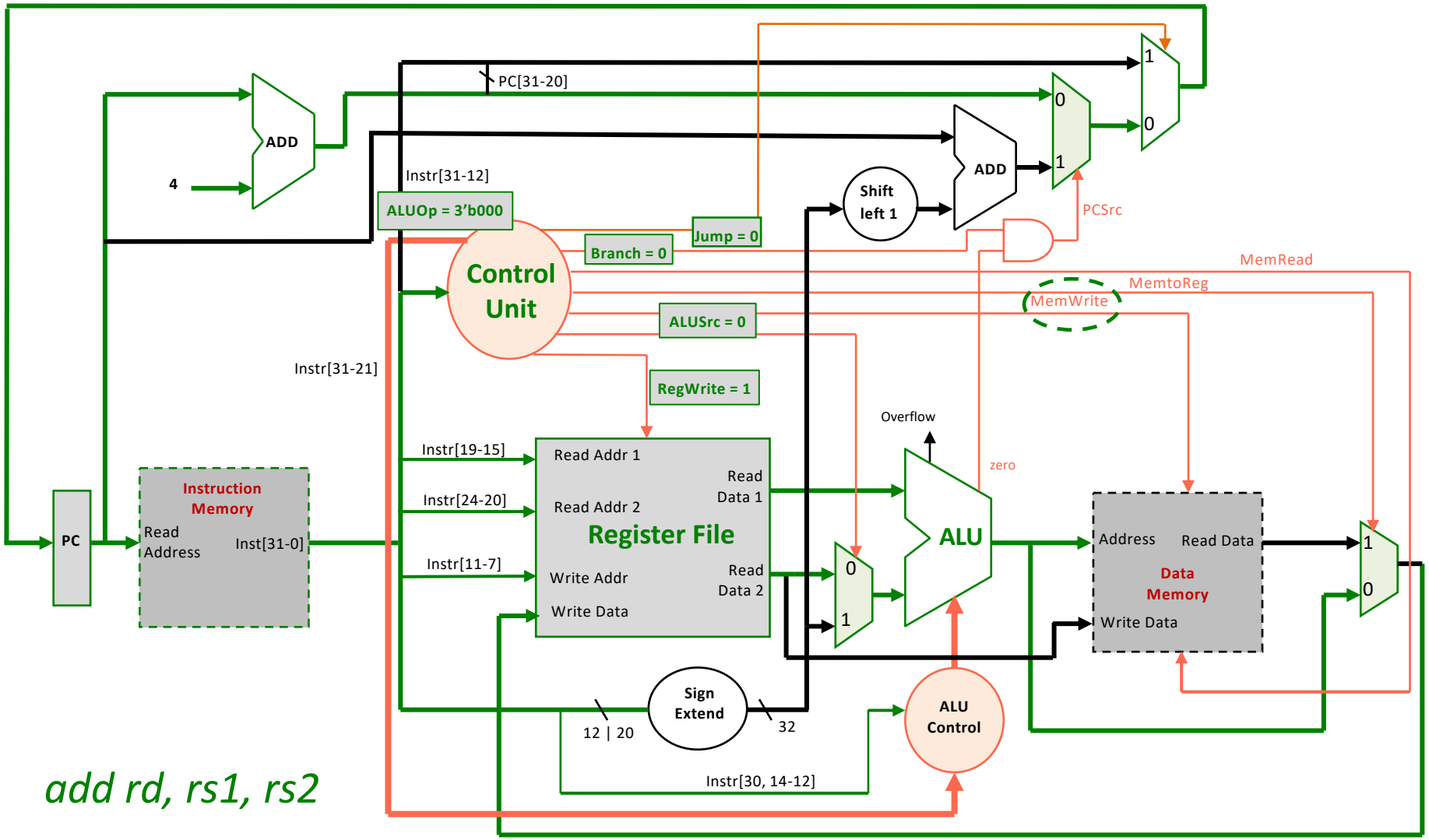
add rd, rs1, rs2

CPU: R-Type Execution Datapath



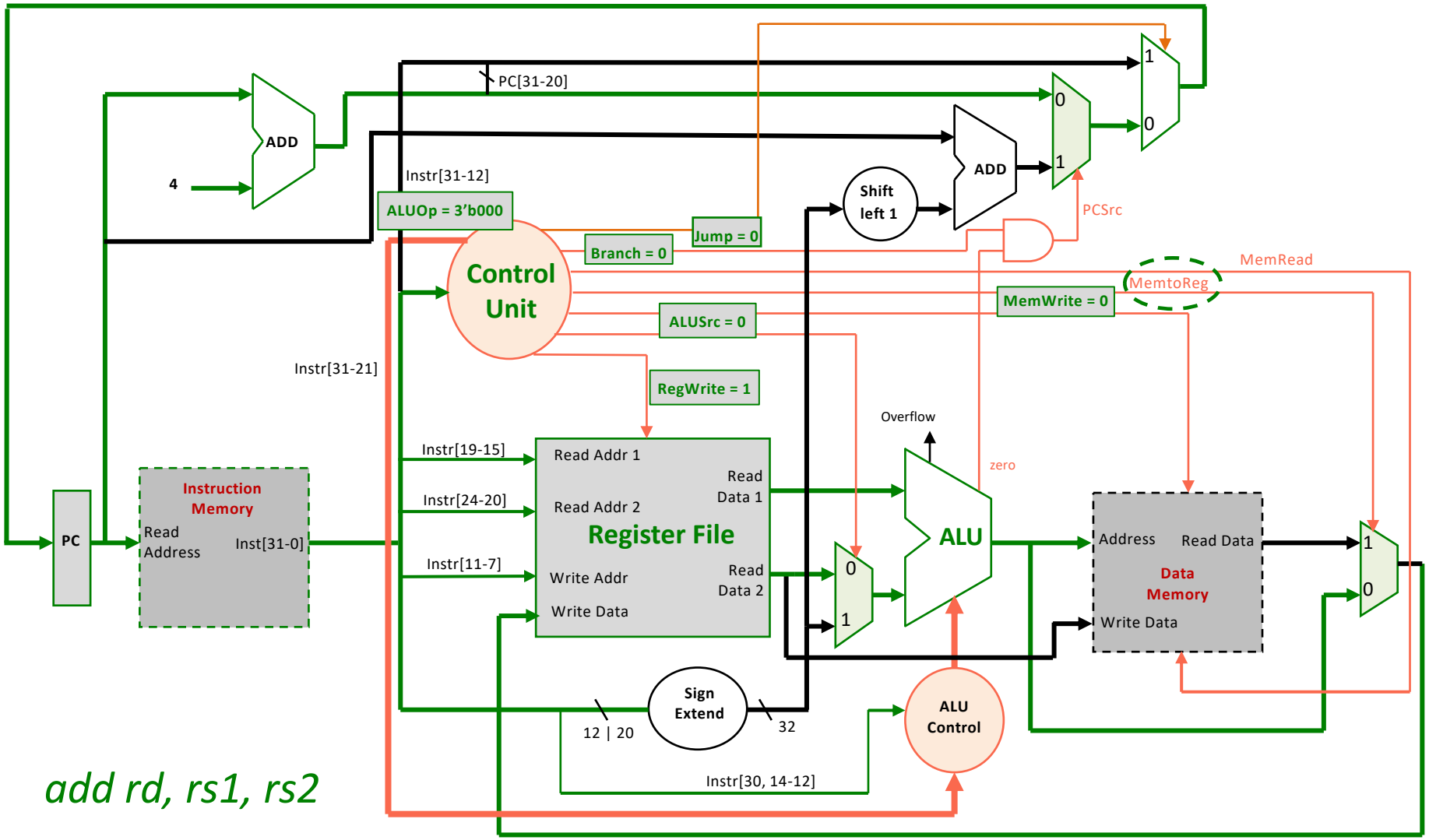
add rd, rs1, rs2

CPU: R-Type Execution Datapath



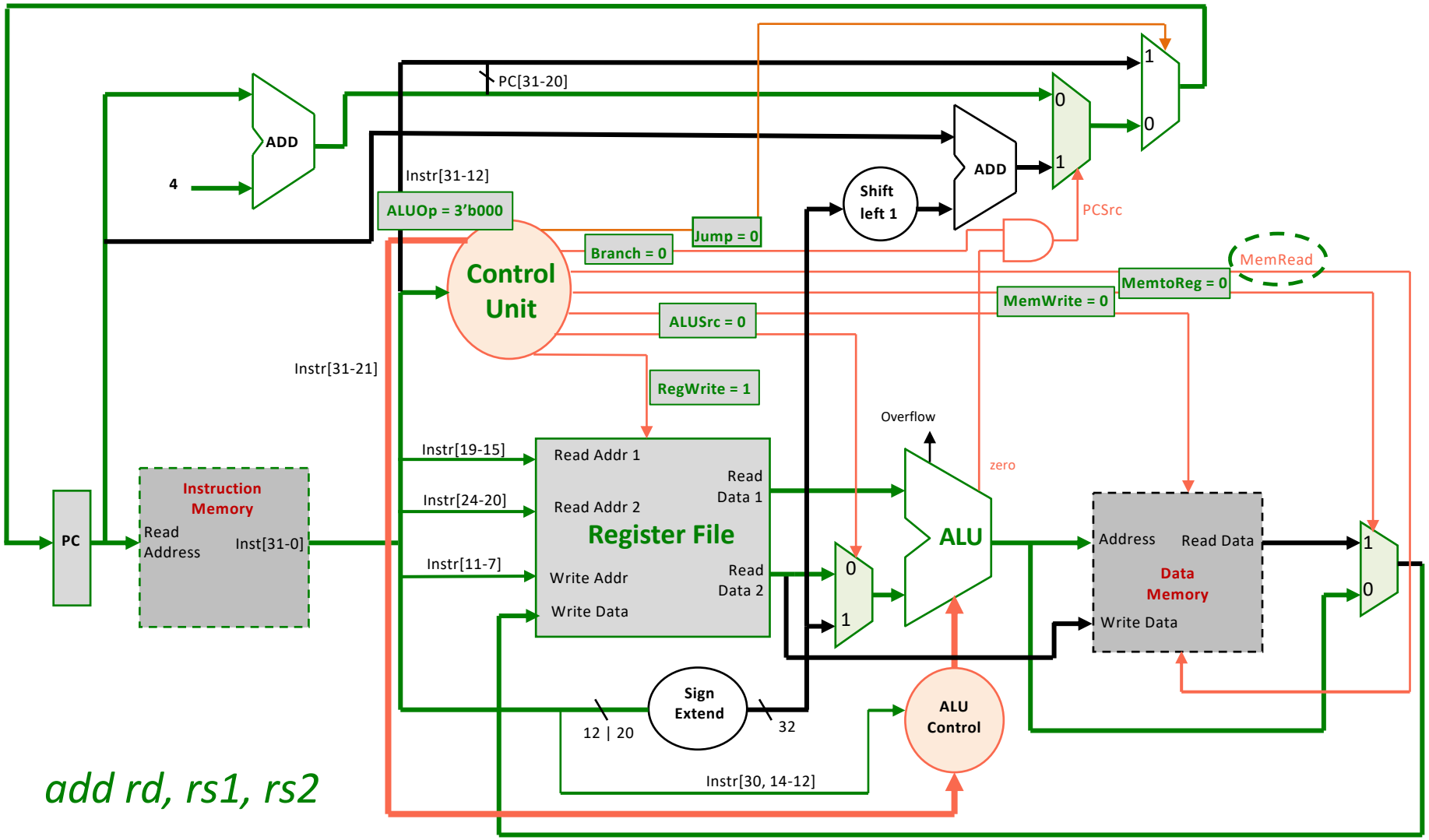
add rd, rs1, rs2

CPU: R-Type Execution Datapath



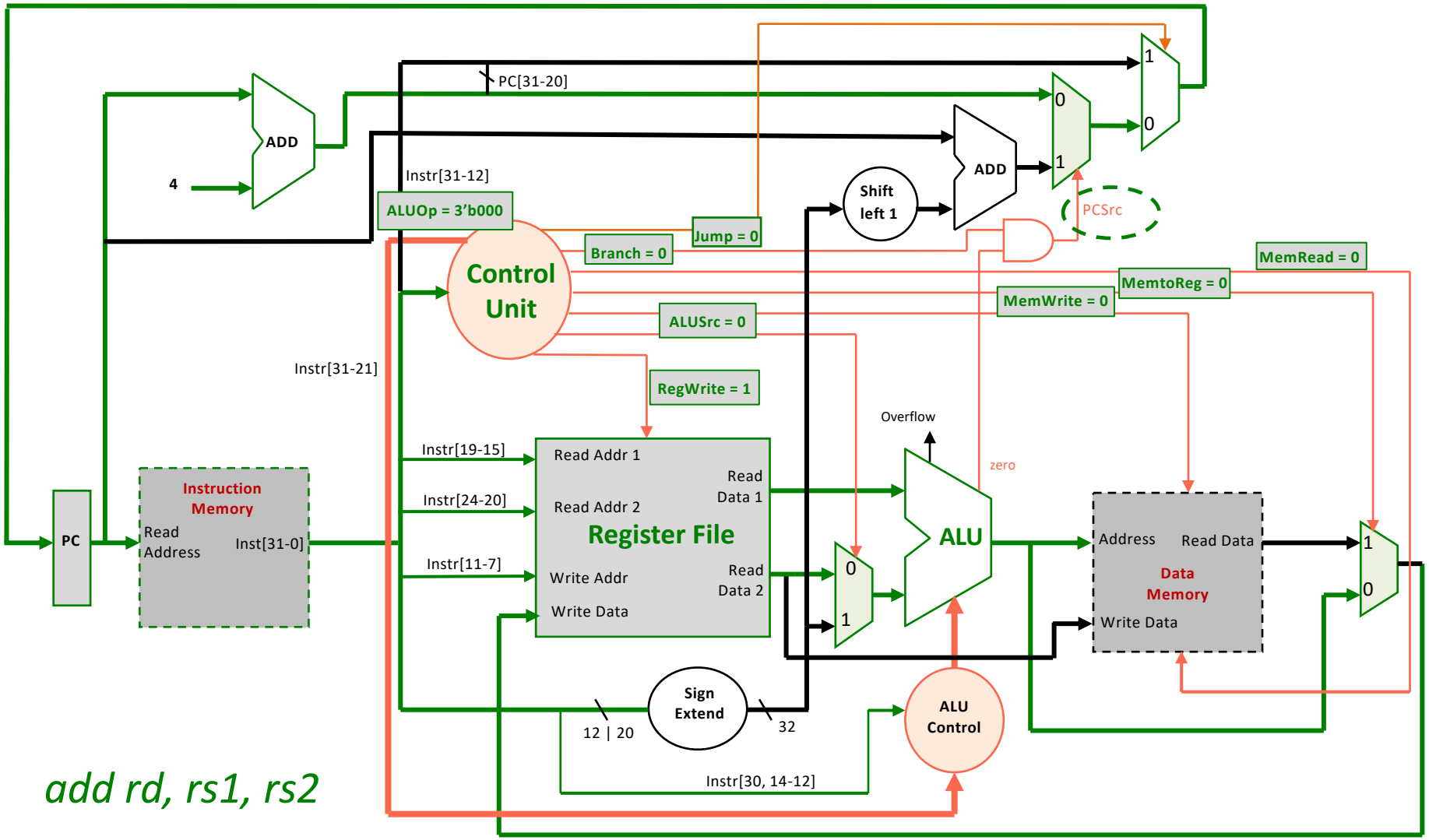
add rd, rs1, rs2

CPU: R-Type Execution Datapath



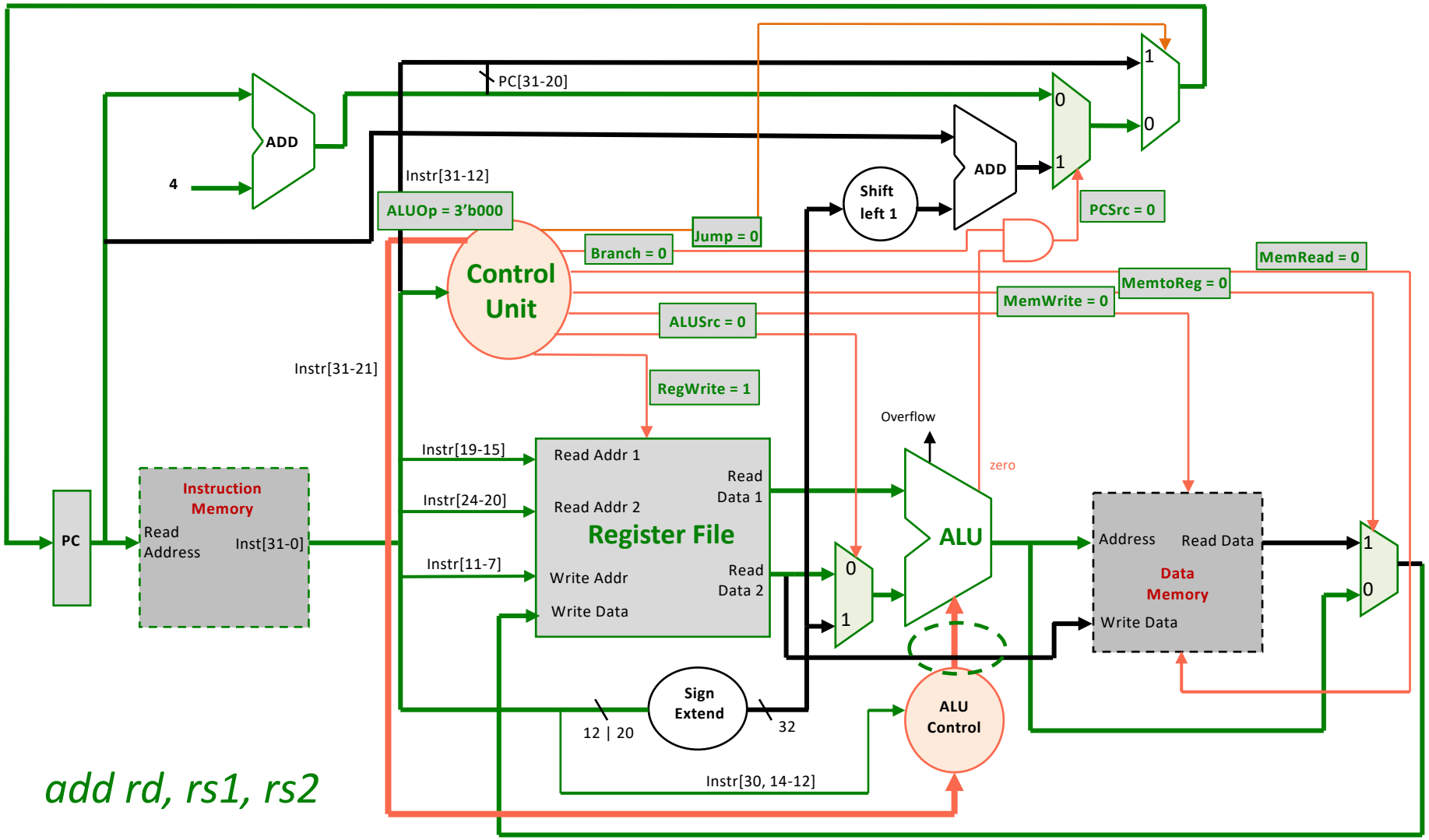
add rd, rs1, rs2

CPU: R-Type Execution Datapath



add rd, rs1, rs2

CPU: R-Type Execution Datapath

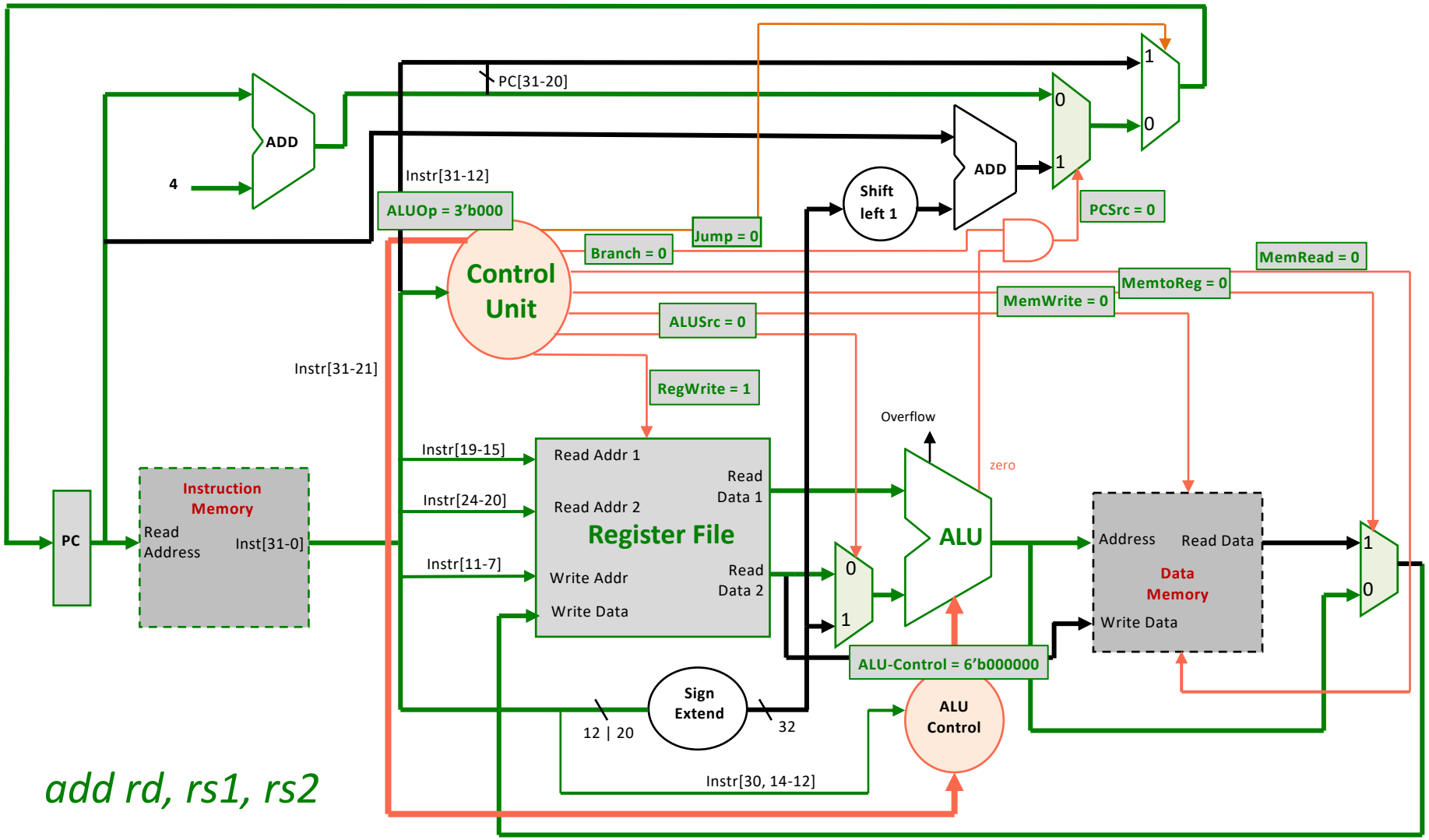


add rd, rs1, rs2

ALU Operation

Instruction Opcode	Instruction Operation	ALU Op	funct3	funct7	Effective ALU Operation	ALU Control
LW	Load word	100	010	xxxxxxx	Add	000000
SW	Store word	101	010	xxxxxxx	Add	000000
BEQ	Branch equal	010	000	xxxxxxx	Subtract	001000
R-type	ADD	000	000	0000000	Add	000000
R-type	SUB	000	000	0100000	Subtract	001000
R-type	AND	000	111	0000000	and	000111
R-type	OR	000	110	0000000	Or	000110
R-type	SLT	000	010	0000000	Set-less-than	000010

CPU: R-Type Execution Datapath

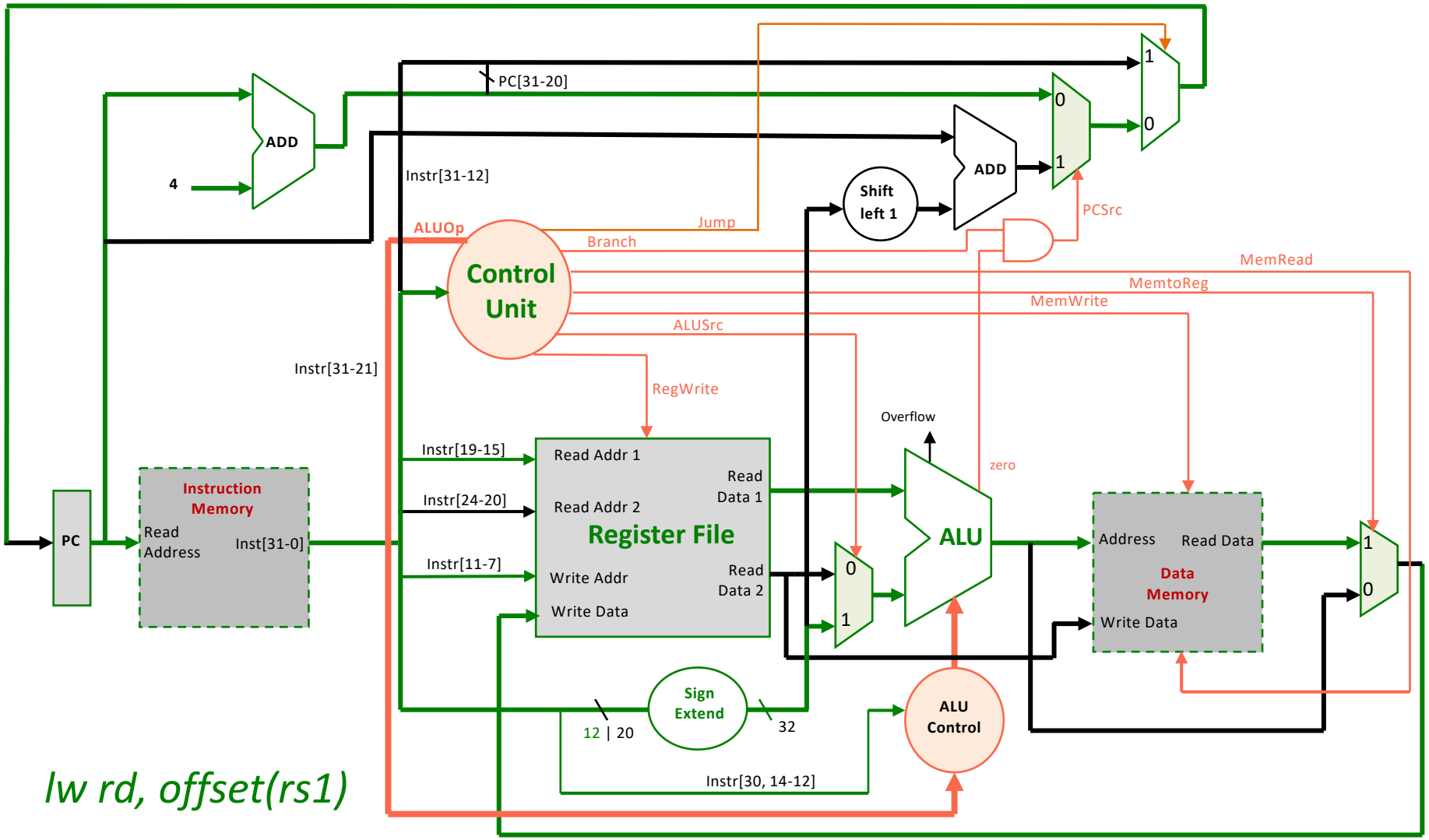


add rd, rs1, rs2

Instruction Execution Flows

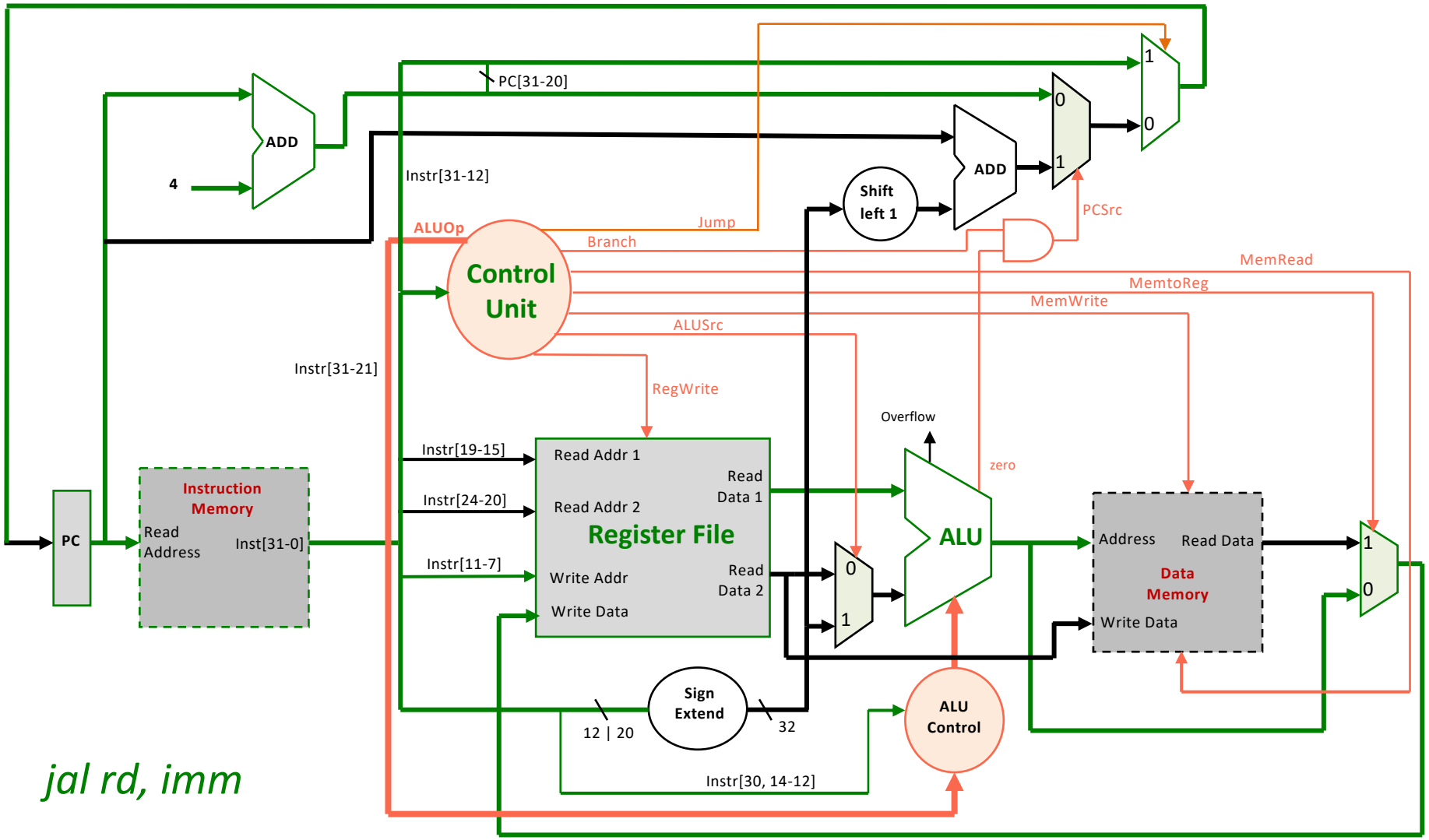
Instruction class	Functional Units used by the instruction class				
R-type	Instruction fetch	Register access	ALU	Register access	
Load word	Instruction fetch	Register access	ALU	Memory access	Register access

CPU: I-Type Execution Datapath



lw rd, offset(rs1)

CPU: U-Type Execution Datapath



jal rd, imm

Instruction Execution Flows

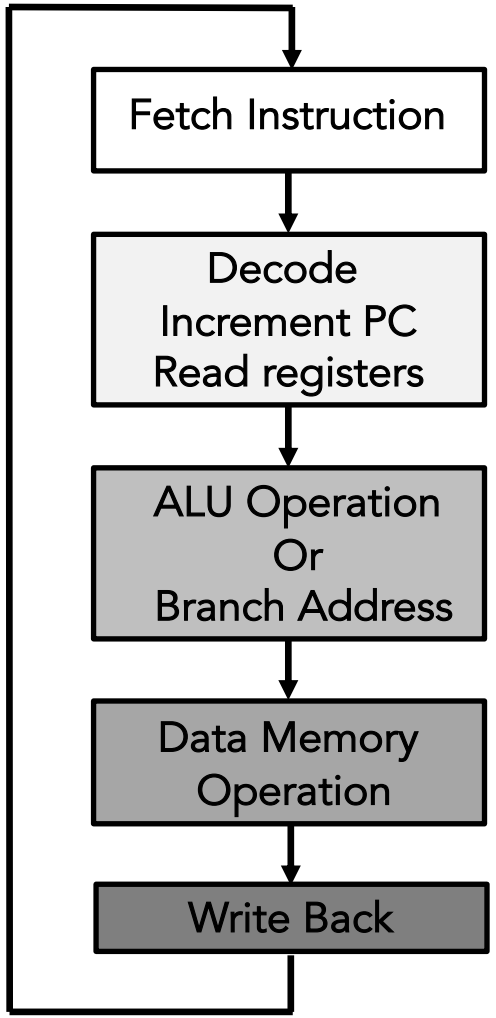
Instruction class	Functional Units used by the instruction class				
R-type	Instruction fetch	Register access	ALU	Register access	
Load word	Instruction fetch	Register access	ALU	Memory access	Register access
Store word	Instruction fetch	Register access	ALU	Memory access	
Branch	Instruction fetch	Register access	ALU		
Jump	Instruction fetch				

CPU Control

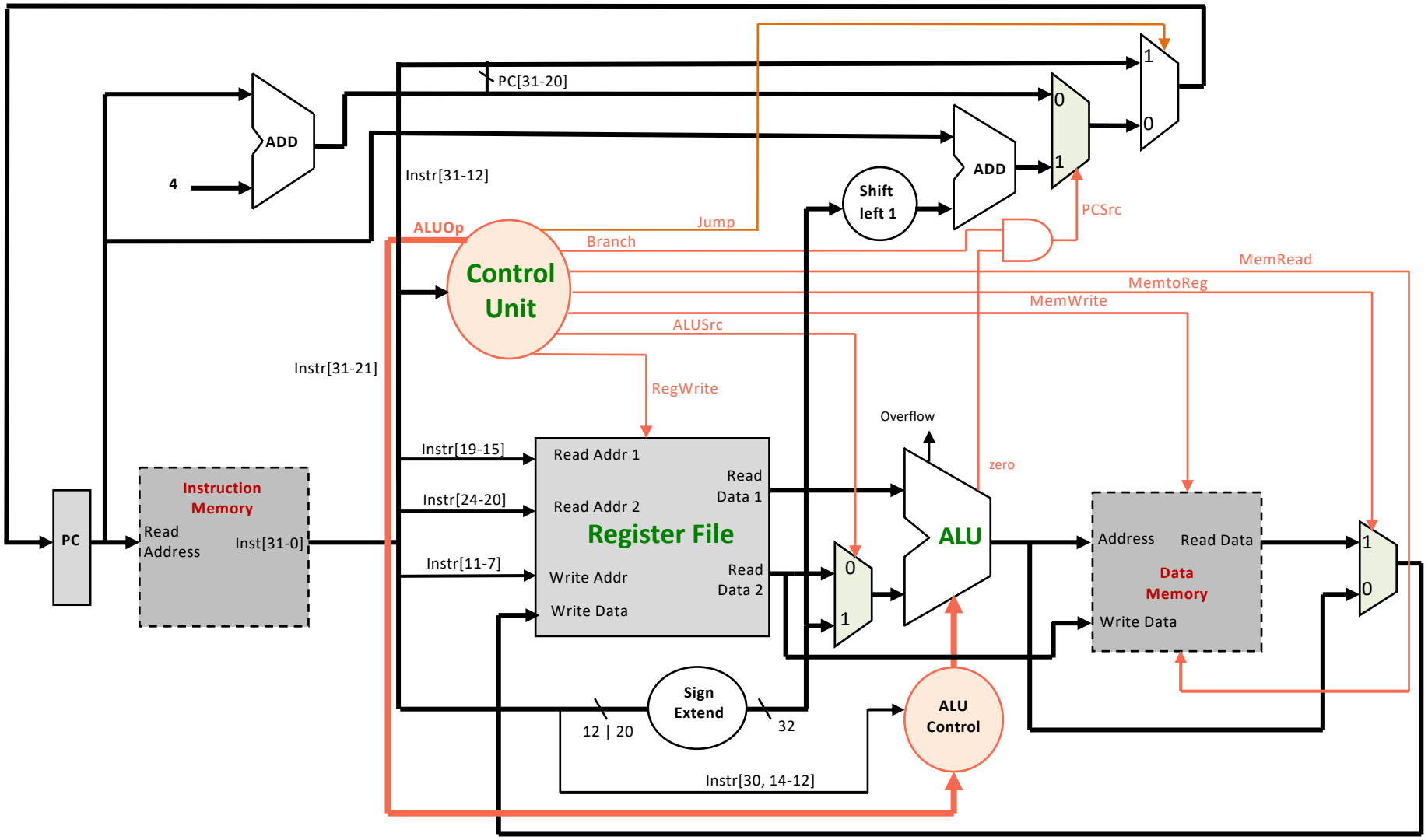
Instruction	RegDest	ALUSrc	Memto-Reg	RegWrite	Mem Read	Mem Write	Branch
R-format	1	0	0	1	0	0	0
lw	0	1	1	1	1	0	0
sw	X	1	X	0	0	1	0
beq	X	0	X	0	0	0	1

Central Processing Unit (CPU)

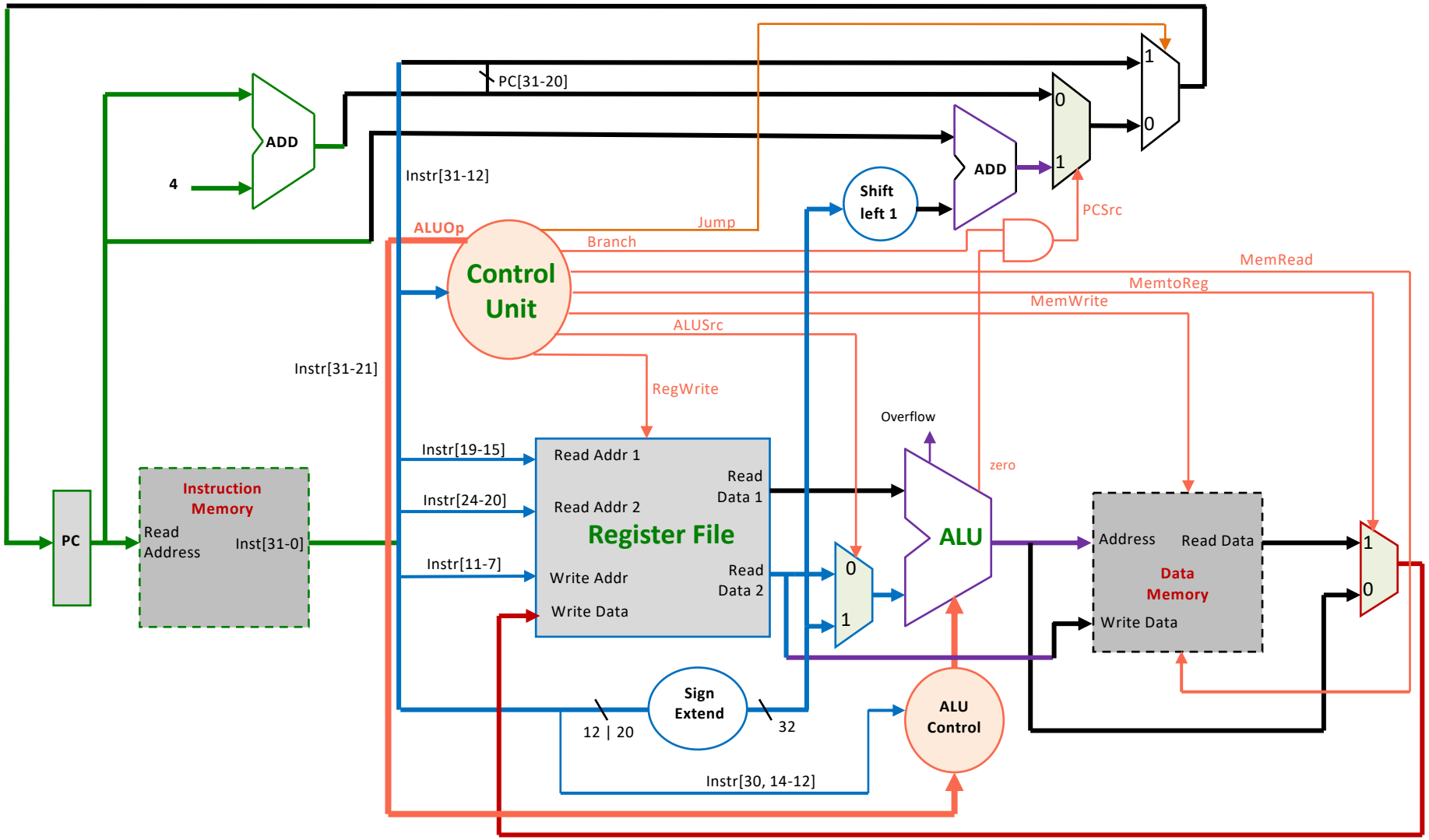
- Central Processing Unit (CPU) Organization
- CPU Execution Process
  1. Fetch Instruction
  2. Decode Instruction
  3. Execute Operation
  4. Memory Operation
  5. Register Writeback Operation



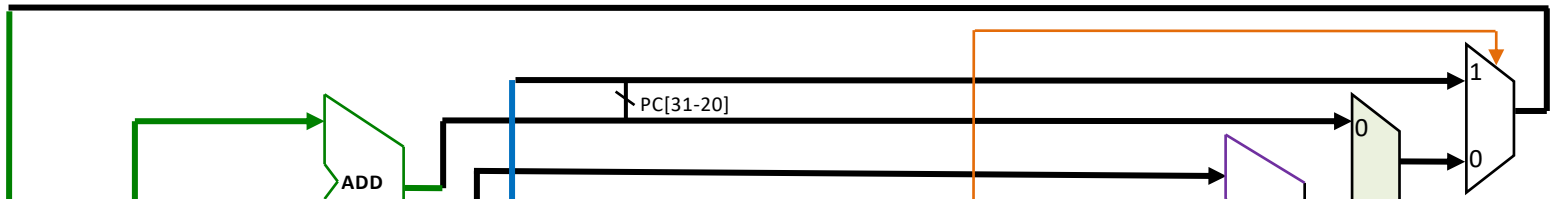
Single Cycle RISC-V CPU

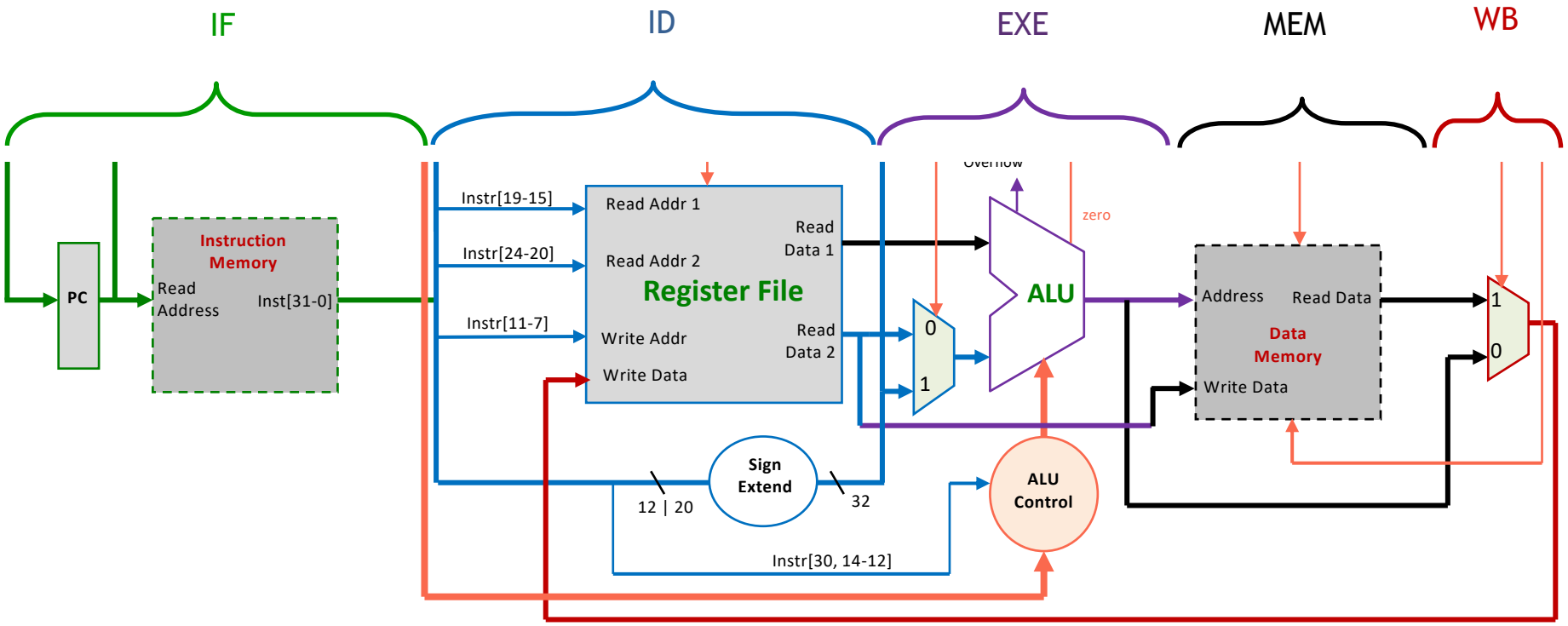


Single Cycle RISC-V CPU

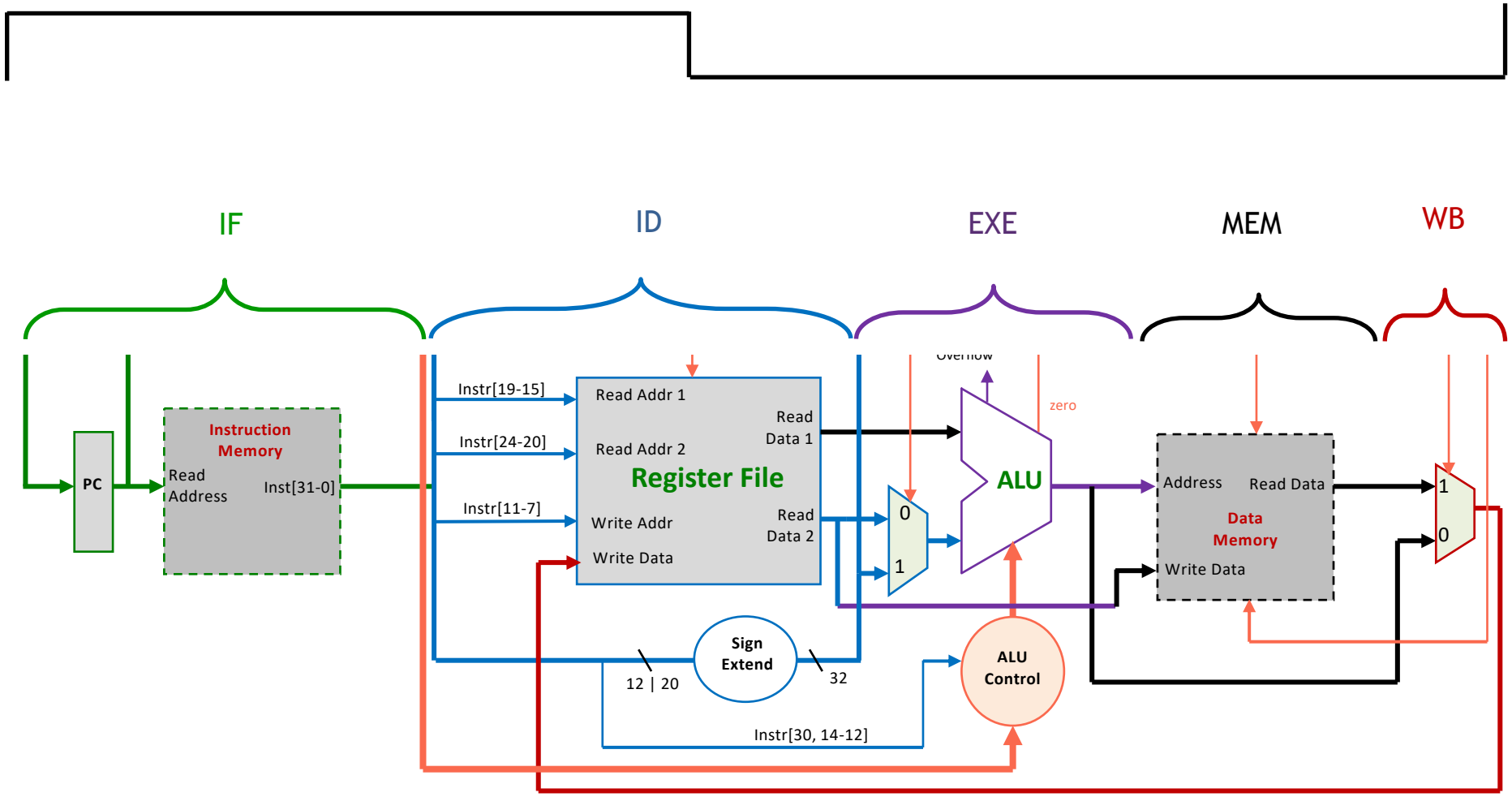


Single Cycle RISC-V CPU

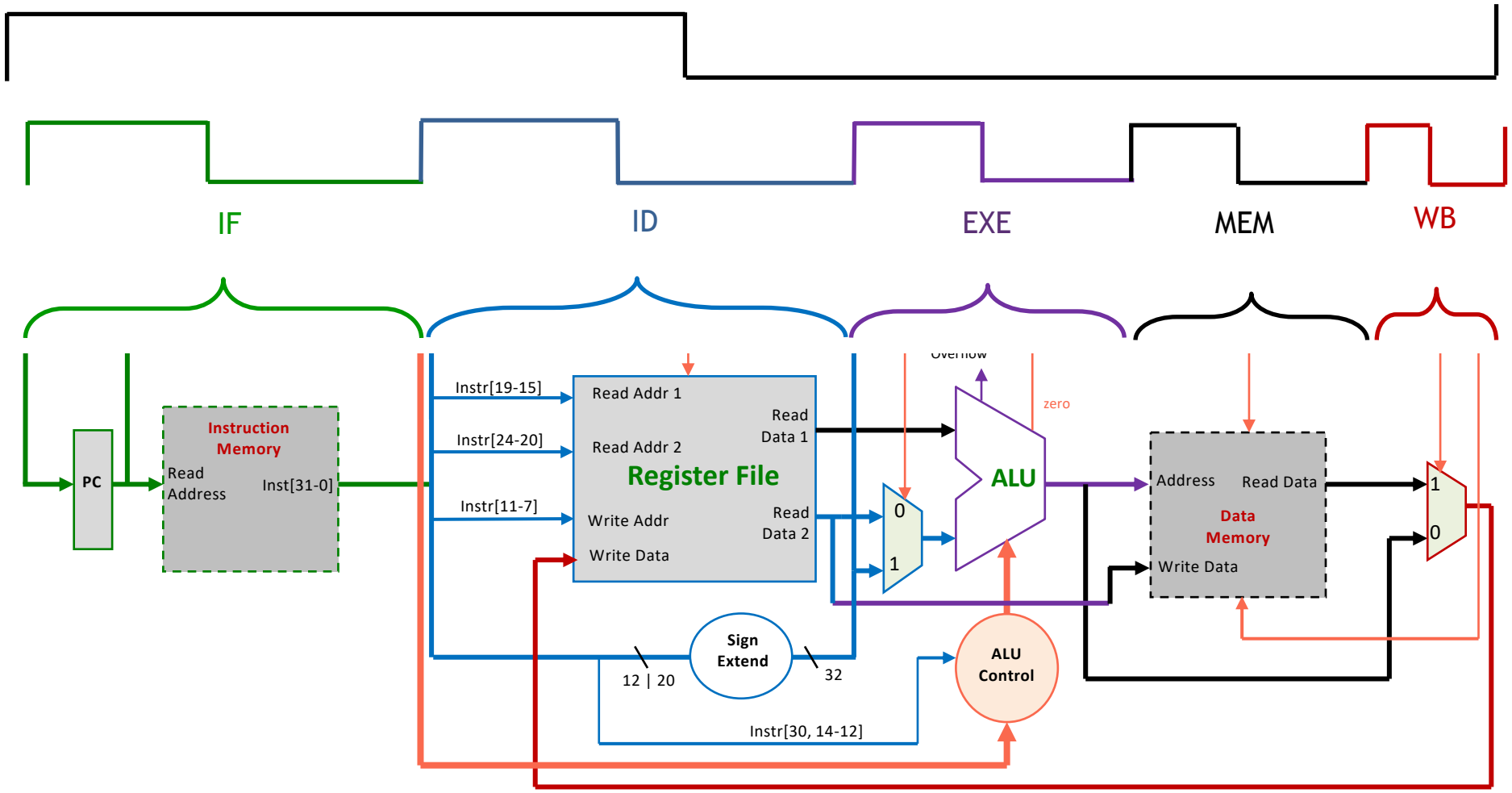




Single Cycle RISC-V CPU



Single Cycle RISC-V CPU



Multi-Cycle RISC-V Processor

- Single cycle processor
  - The cycle time has to be long enough for the slowest instruction to complete
  - CPI = 1
- Multi-cycle Implementation
  - Instructions are broken into smaller steps
  - Execute each step (instead of the entire instruction) in one cycle
  - The cycle time has to be long enough for the slowest step to complete

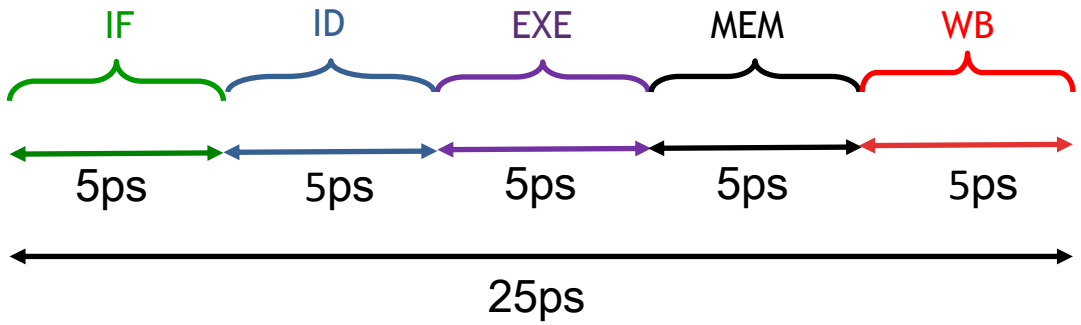
Multi-Cycle RISC-V Processor

- The advantages of the multiple cycle processor
  - Cycle time is much shorter
  - Different instructions take different number of cycles to complete
    - Load takes five cycles
    - Jump only takes three cycles
    - CPI is between 3 and 5

```
addi   x10, x10, 413
beq    x11, x10, label
```

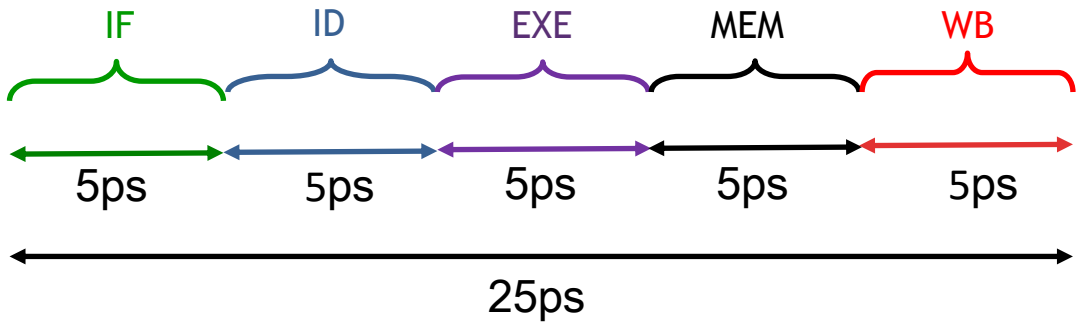
Multi-Cycle RISC-V Processor

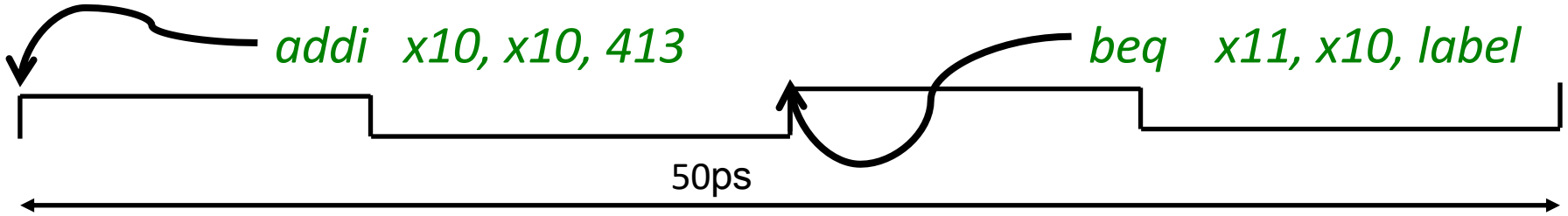
addi x10, x10, 413
beq x11, x10, label



Multi-Cycle RISC-V Processor

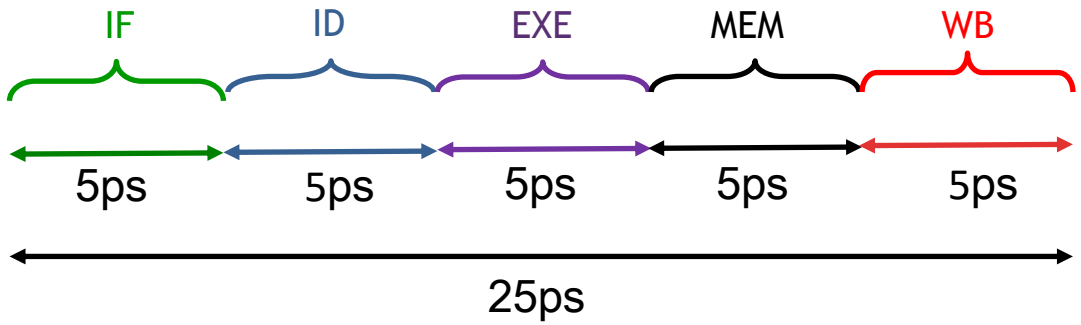
addi x10, x10, 413
beq x11, x10, label

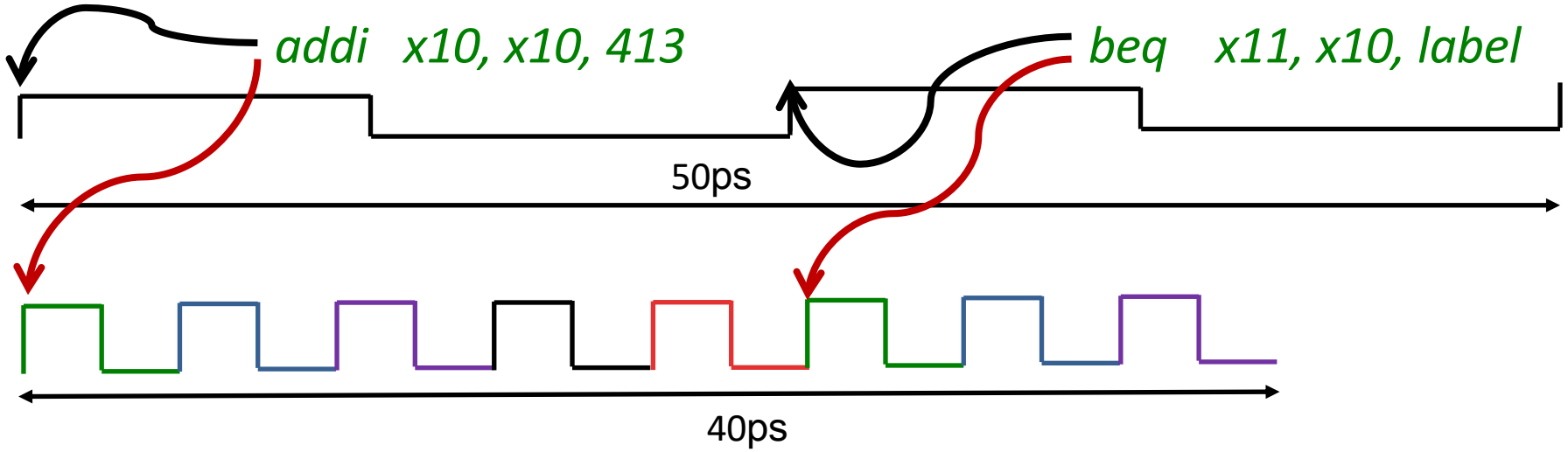




Multi-Cycle RISC-V Processor

addi x10, x10, 413
beq x11, x10, label





Multi-Cycle RISC-V Processor

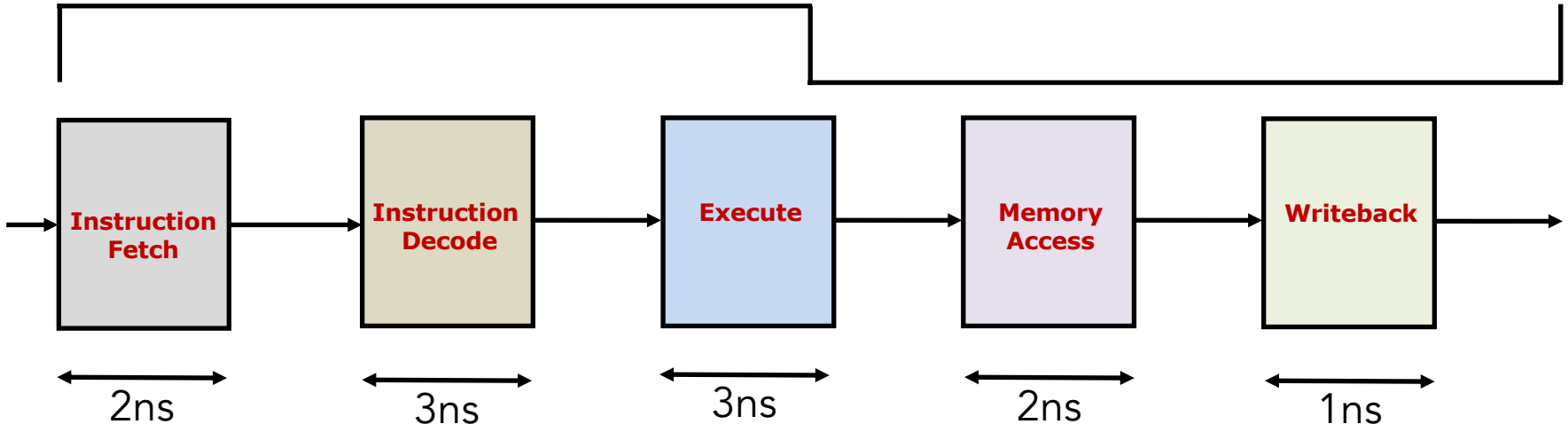
- Although CPI increased overall execution time is shortened, so performance is improved!

$$\frac{\text{Time}}{\text{Program}} = \frac{\text{Instructions}}{\text{Program}} * \frac{\text{Cycles}}{\text{Instruction}} * \frac{\text{Time}}{\text{Cycle}}$$

- Performance can be improved even more with pipelining!

RISC-V CPU Stages

- Single Cycle CPU Instruction Execution

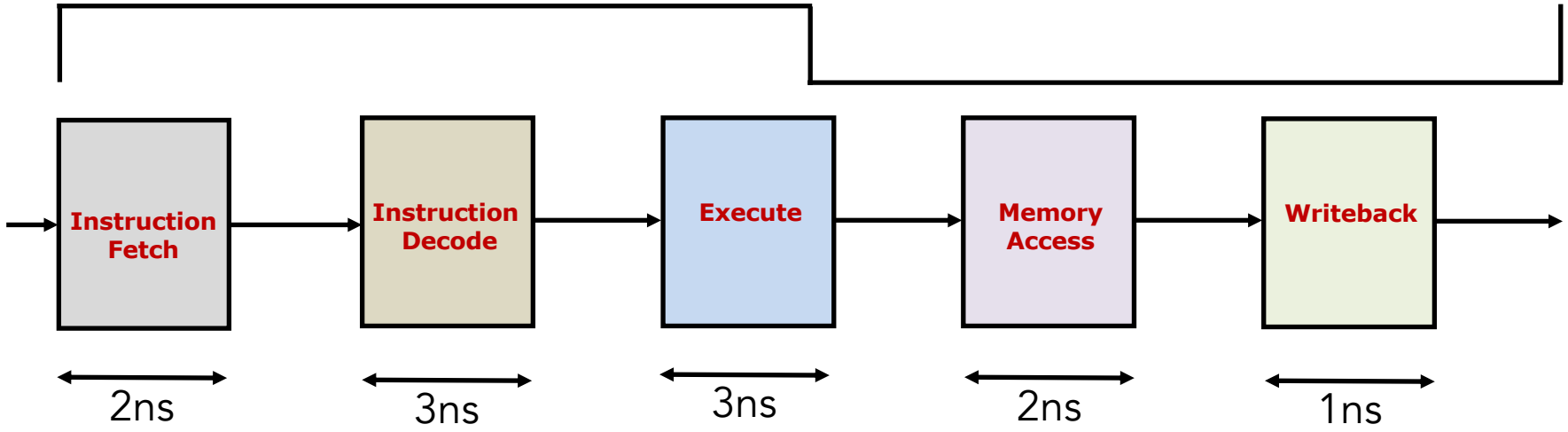


Cycle Time: 2ns + 3ns + 3ns + 2ns + 1ns = 11ns

```
I1      addi x3,x0,12
Loop:
I2      lw   x1,4(x2)
I3      slli x5,x1,2
I4      add  x3,x3,x3
I5      bge  x6,x5,Loop
I6      addi x5,x5,-1
I7      j    Exit
Exit:
I8            addi x5,x0,8
```

RISC-V CPU Stages

- Single Cycle CPU Instruction Execution



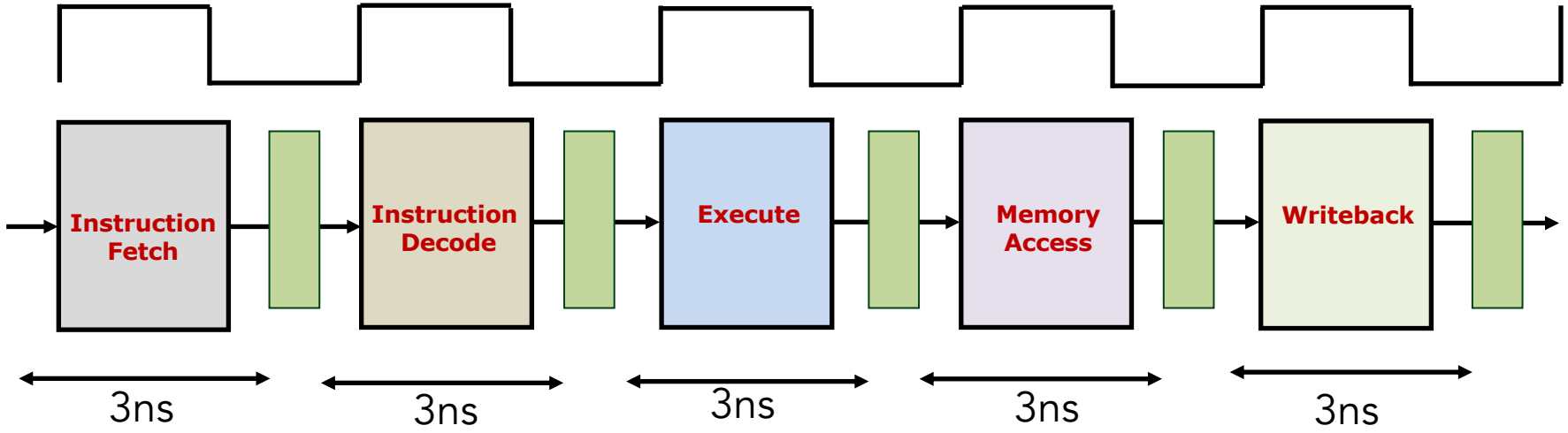
Cycle Time: 2ns + 3ns + 3ns + 2ns + 1ns = 11ns

Total Execution Time: 8 instruction * 11ns = 88 ns

```
I1      addi x3,x0,12
Loop:
I2      lw   x1,4(x2)
I3      slli x5,x1,2
I4      add  x3,x3,x3
I5      bge  x6,x5,Loop
I6      addi x5,x5,-1
I7      j    Exit
Exit:
I8            addi x5,x0,8
```

RISC-V CPU Stages

- Multi-Cycle CPU Instruction Execution

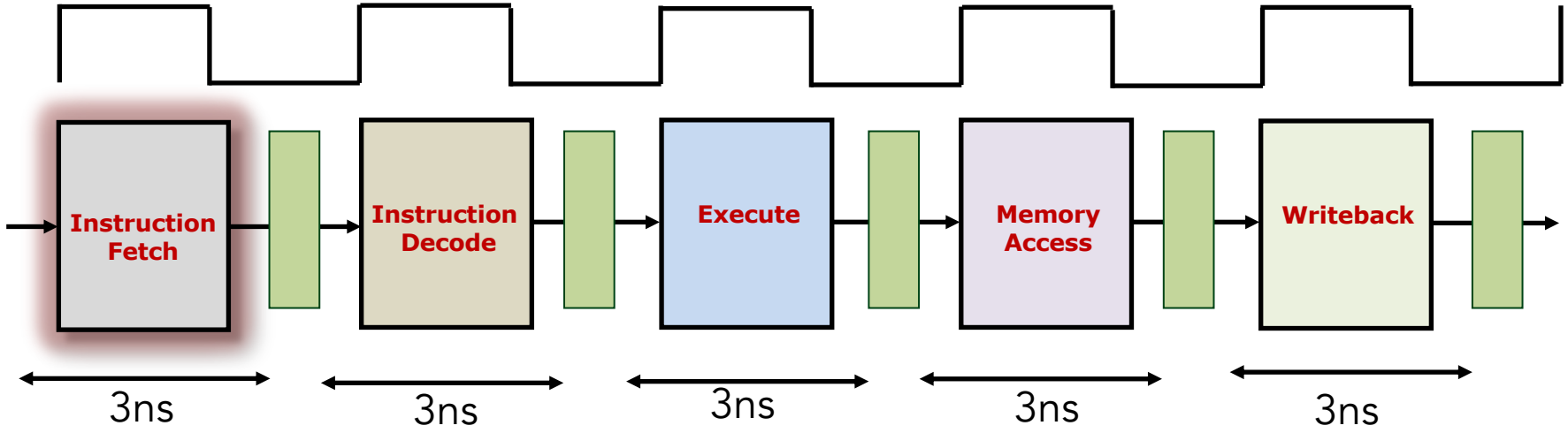


Cycle Time: Max(2ns, 3ns, 3ns, 2ns, 1ns) = 3ns

```
I1        addi x3,x0,12
Loop:
I2        lw   x1,4(x2)
I3        slli x5,x1,2
I4        add  x3,x3,x3
I5        bge  x6,x5,Loop
I6        addi x5,x5,-1
I7        j    Exit
Exit:
I8              addi x5,x0,8
```

RISC-V CPU Stages

- Multi-Cycle CPU Instruction Execution



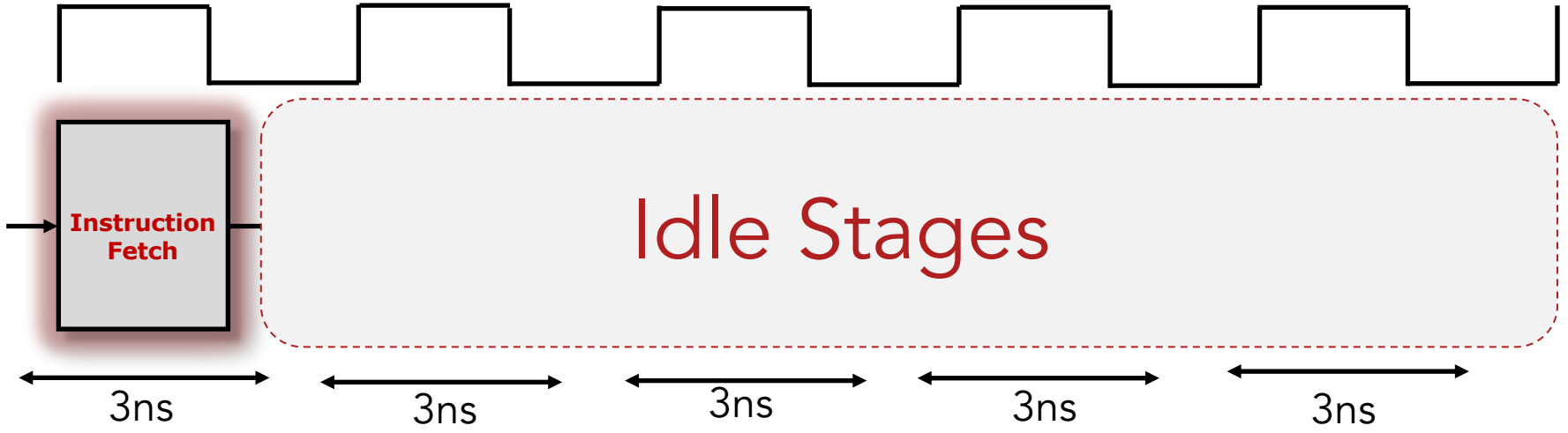
I_1 `addi x3,x0,12`

Cycle Time: Max(2ns, 3ns, 3ns, 2ns, 1ns) = 3ns

```
I1        addi x3,x0,12
Loop:
I2        lw   x1,4(x2)
I3        slli x5,x1,2
I4        add  x3,x3,x3
I5        bge  x6,x5,Loop
I6        addi x5,x5,-1
I7        j    Exit
Exit:
I8              addi x5,x0,8
```

RISC-V CPU Stages

- Multi-Cycle CPU Instruction Execution



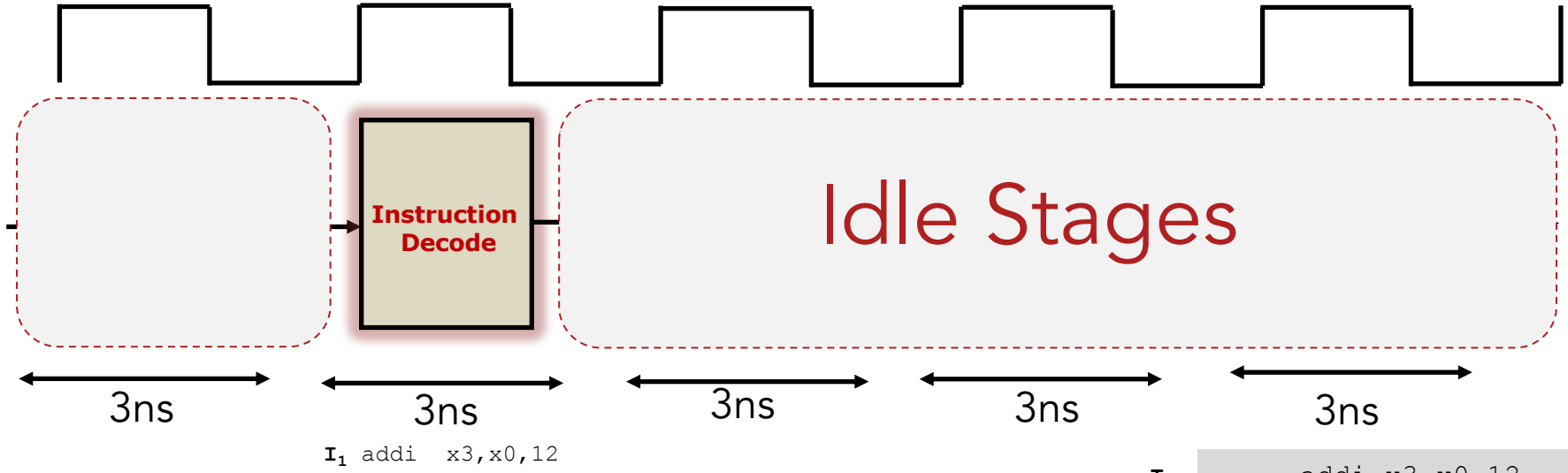
I_1 `addi x3,x0,12`

Cycle Time: Max(2ns, 3ns, 3ns, 2ns, 1ns) = 3ns

```
I1      addi x3,x0,12
Loop:
I2      lw   x1,4(x2)
I3      slli x5,x1,2
I4      add  x3,x3,x3
I5      bge  x6,x5,Loop
I6      addi x5,x5,-1
I7      j    Exit
Exit:
I8           addi x5,x0,8
```

RISC-V CPU Stages

- Multi-Cycle CPU Instruction Execution



Cycle Time: Max(2ns, 3ns, 3ns, 2ns, 1ns) = 3ns

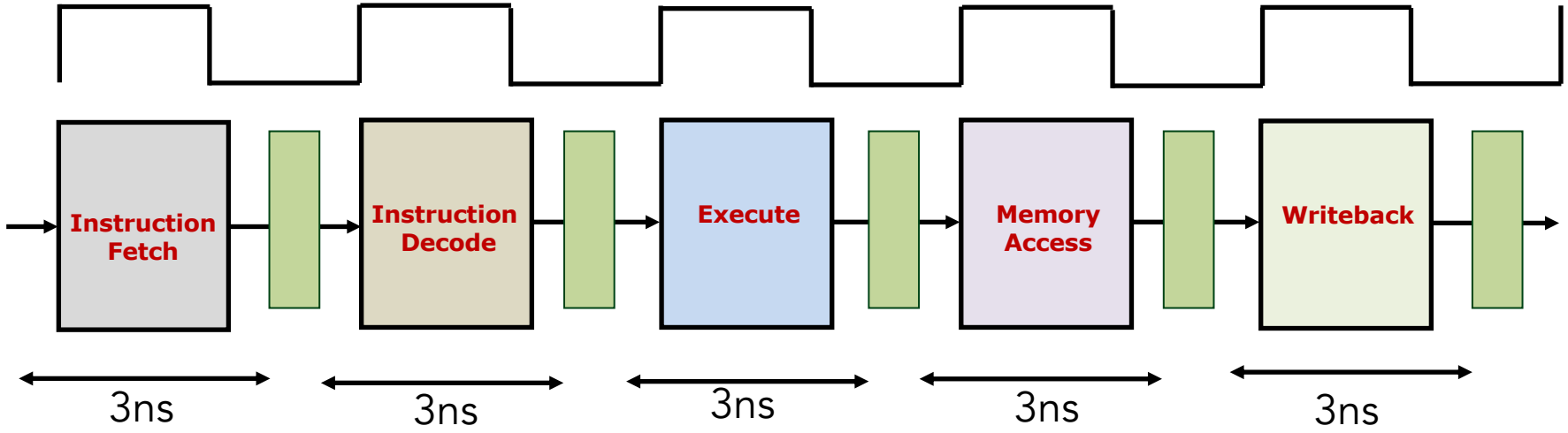
```
I1        addi x3,x0,12
Loop:
I2        lw   x1,4(x2)
I3        slli x5,x1,2
I4        add  x3,x3,x3
I5        bge  x6,x5,Loop
I6        addi x5,x5,-1
I7        j    Exit
Exit:
I8              addi x5,x0,8
```

Instruction Execution Stages

Instruction class	CPU stages used by the instruction class				
R-type	Instruction fetch	Instruction Decode	Execute	Writeback	
Load word	Instruction fetch	Instruction Decode	Execute	Memory Access	Writeback
Store word	Instruction fetch	Instruction Decode	Execute	Memory Access	
Branch	Instruction fetch	Instruction Decode	Execute		
Jump	Instruction fetch	Instruction Decode			

RISC-V CPU Stages

- Multi-Cycle CPU Instruction Execution



Cycle Time: Max(2ns, 3ns, 3ns, 2ns, 1ns) = 3ns

I_1 , I_3 , I_4 , I_6 , I_8 → 5 Instructions * 4 stages * 3ns = 60 ns

I_2 → 5 stages * 3ns = 15 ns

I_5 → 3 stages * 3ns = 9 ns

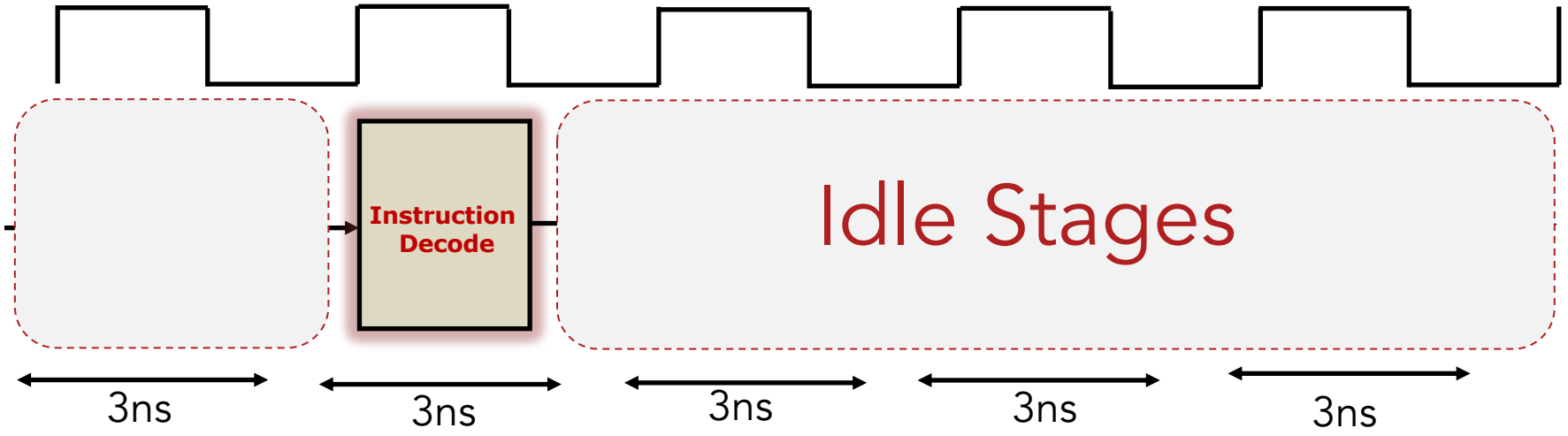
I_7 → 2 stages * 3ns = 6 ns

Total Execution Time: 90 ns

```
I1      addi x3,x0,12
Loop:
I2      lw   x1,4(x2)
I3      slli x5,x1,2
I4      add  x3,x3,x3
I5      bge  x6,x5,Loop
I6      addi x5,x5,-1
I7      j    Exit
Exit:
I8          addi x5,x0,8
```

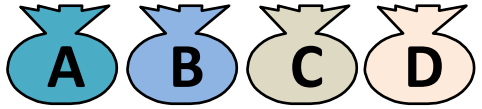
RISC-V CPU Stages

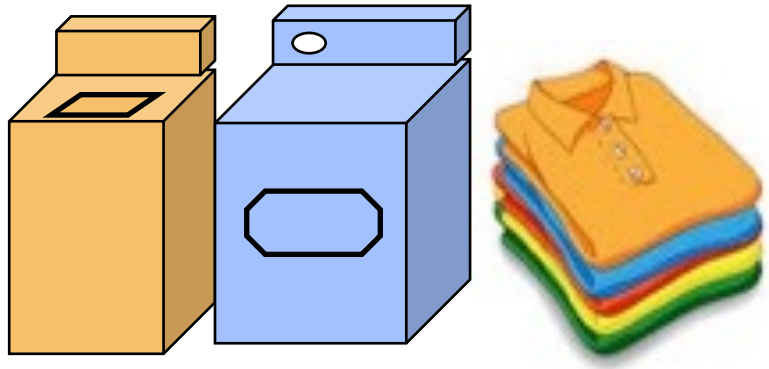
- Issue with Multi-Cycle CPU Instruction Execution



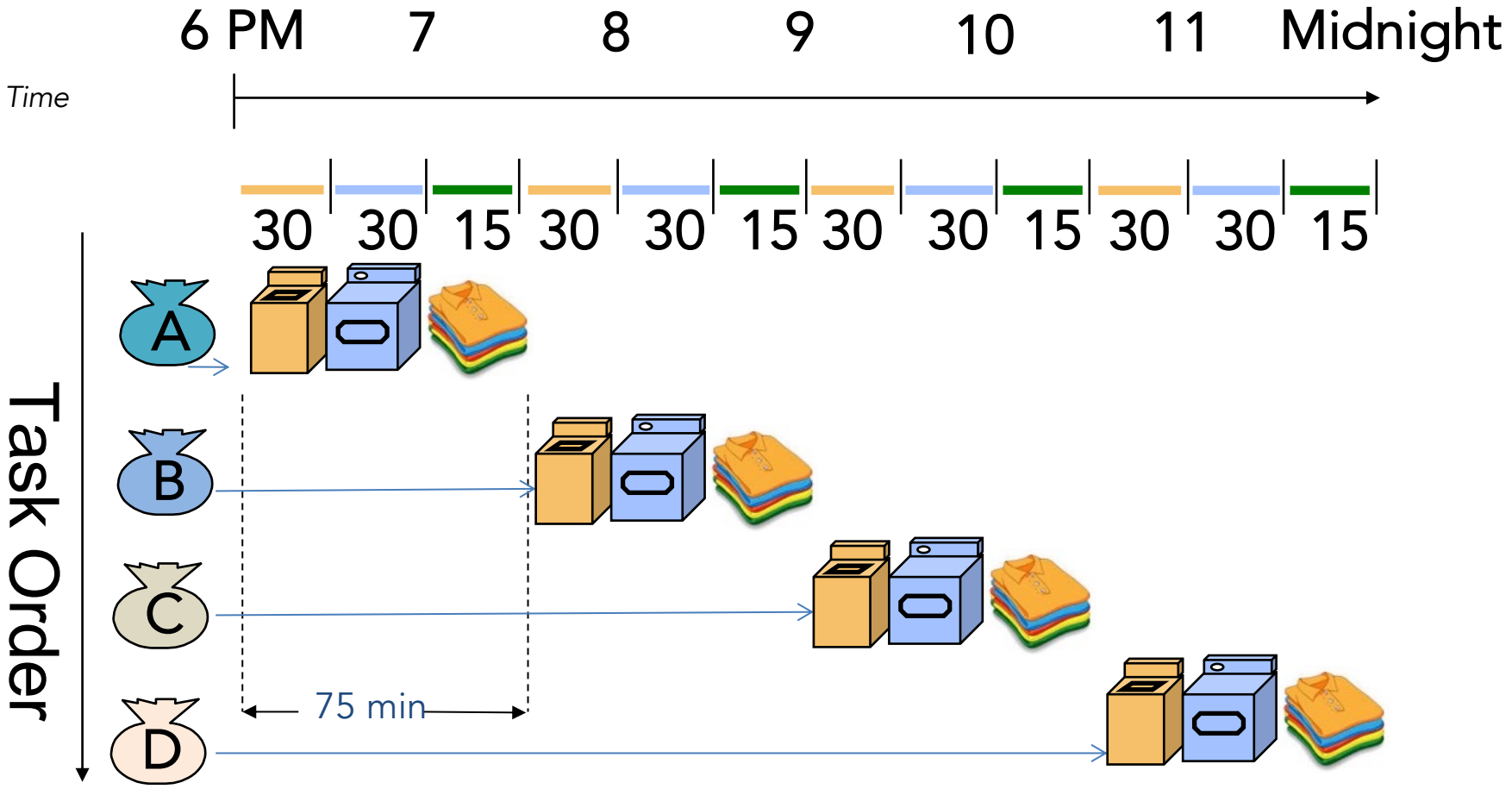
Task Pipelining

- Albert's laundry room has one washer, one dryer and one operator
  - It takes Albert 75 minutes to finish one load
    - The washer takes 30 minutes
    - The dryer takes 30 minutes
    - Folding takes 15 minutes
- Let us image four loads:



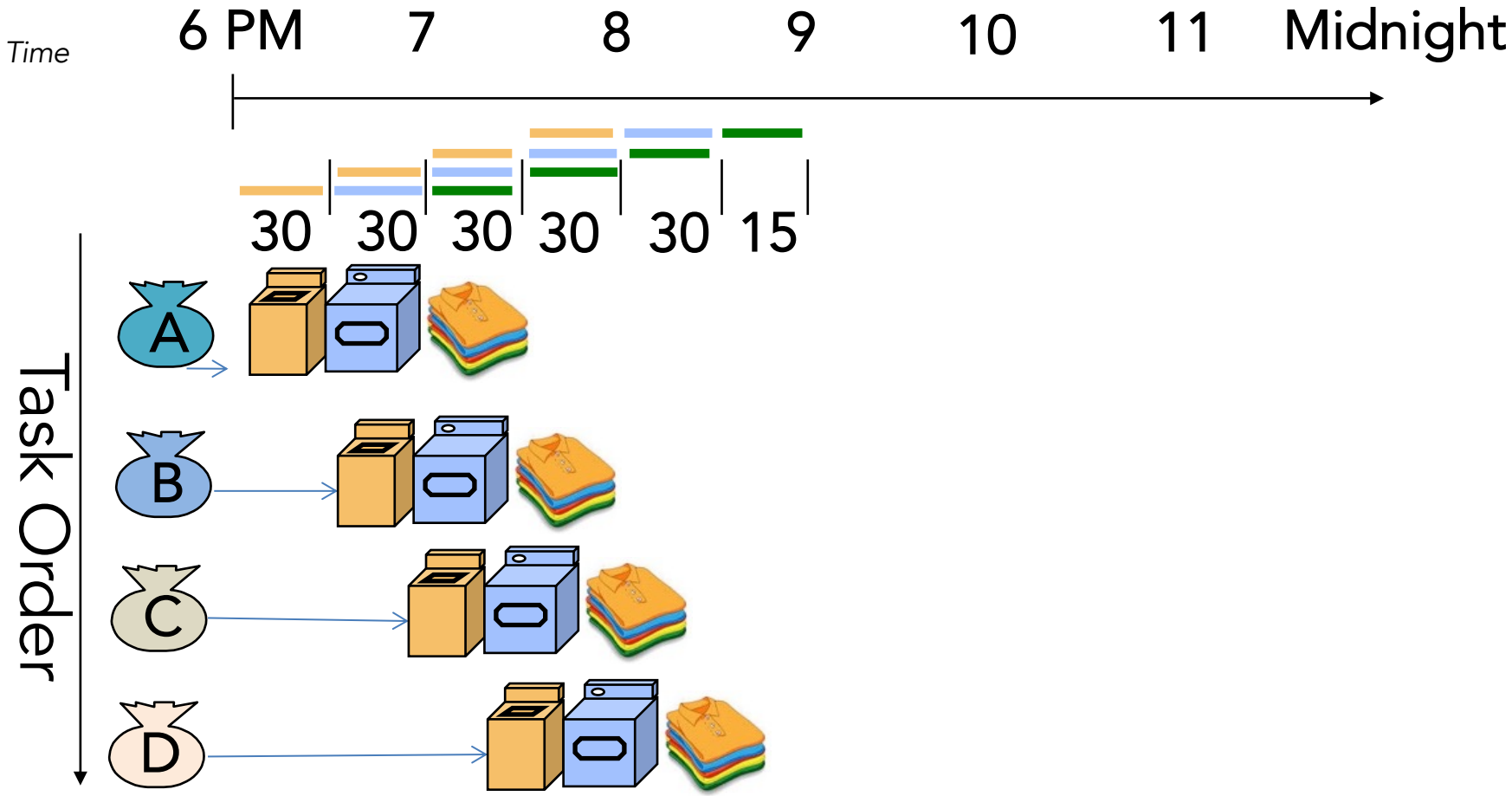


Task Pipelining



- The process is sequential. Sequential laundry takes 5 hours for 4 loads

Task Pipelining



- **Pipelined** laundry takes 2.5 hours for 4 loads

Task Pipelining

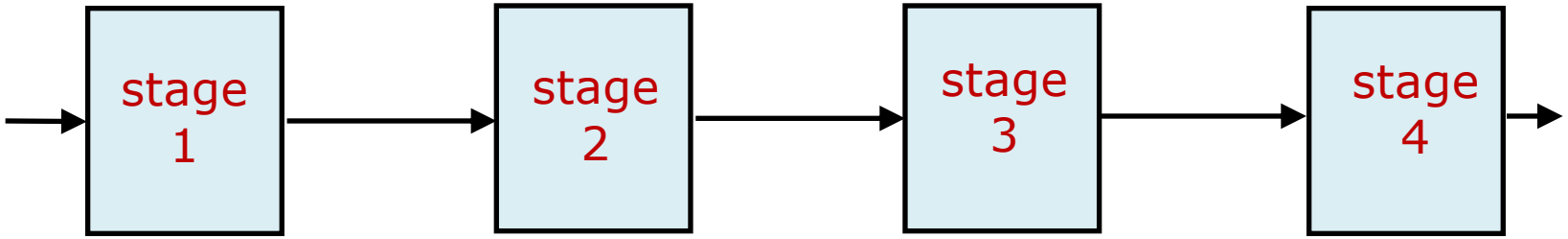
- Time-to-completion: time to completely execute a certain task
  - Laundry load time-to-completion is 75
- Throughput: amount of work that can be done over a period of time
- Pipelining:
  - Does not help time-to-completion for a single task, it helps throughput of the entire system
  - Multiple tasks operating simultaneously using different resources

Task Pipelining

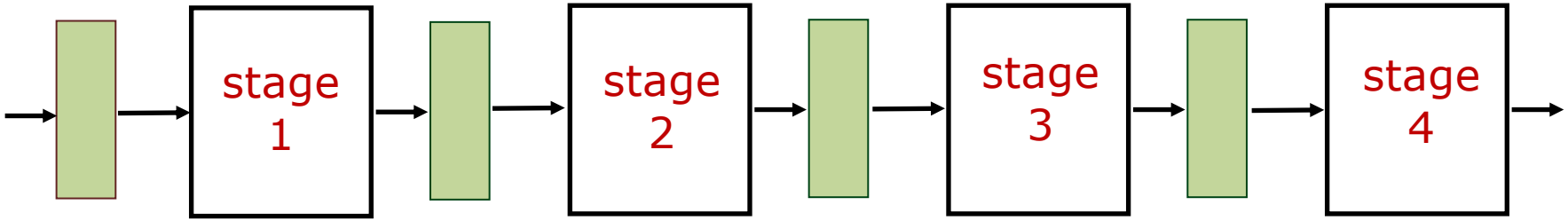
- Time-to-completion: time to completely execute a certain task
  - Laundry load time-to-completion is 75
- Throughput: amount of work that can be done over a period of time
- Pipelining:
  - Potential speedup related to number of stages
  - Time to “fill” pipeline and time to “drain” it reduces speedup

Task Pipelining

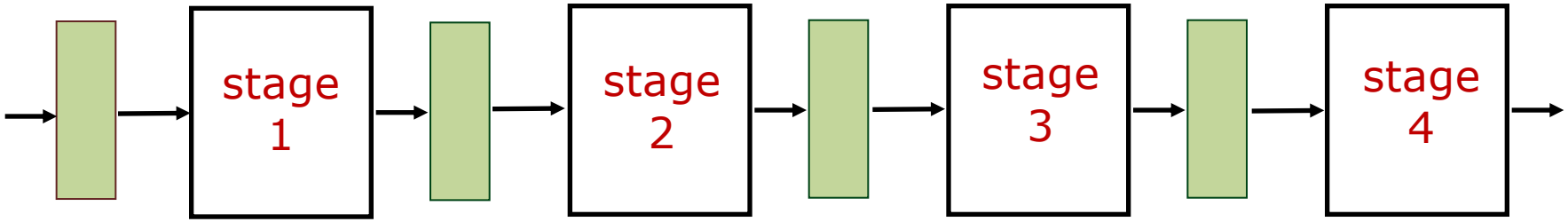
- How to move from stage to stage?



- Buffers/ Registers

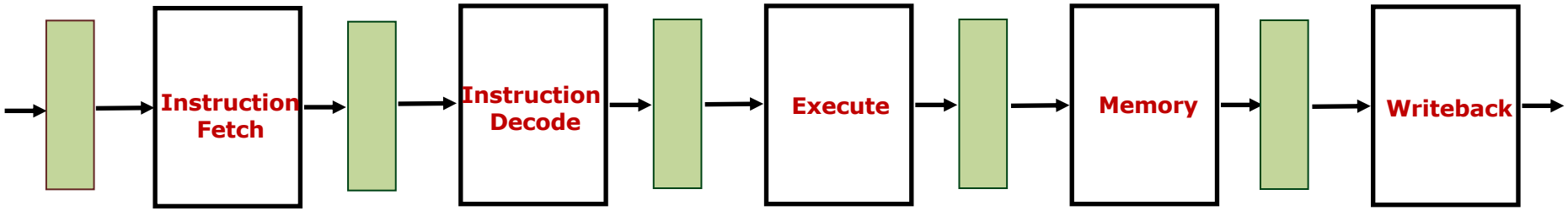


Ideal Pipeline



- All objects go through the same stages
- No sharing of resources between any two stages
- Propagation delay through all pipeline stages is equal

5-Stage RISC-V Pipelining



Next Review Module

- Structural, Data and control hazards