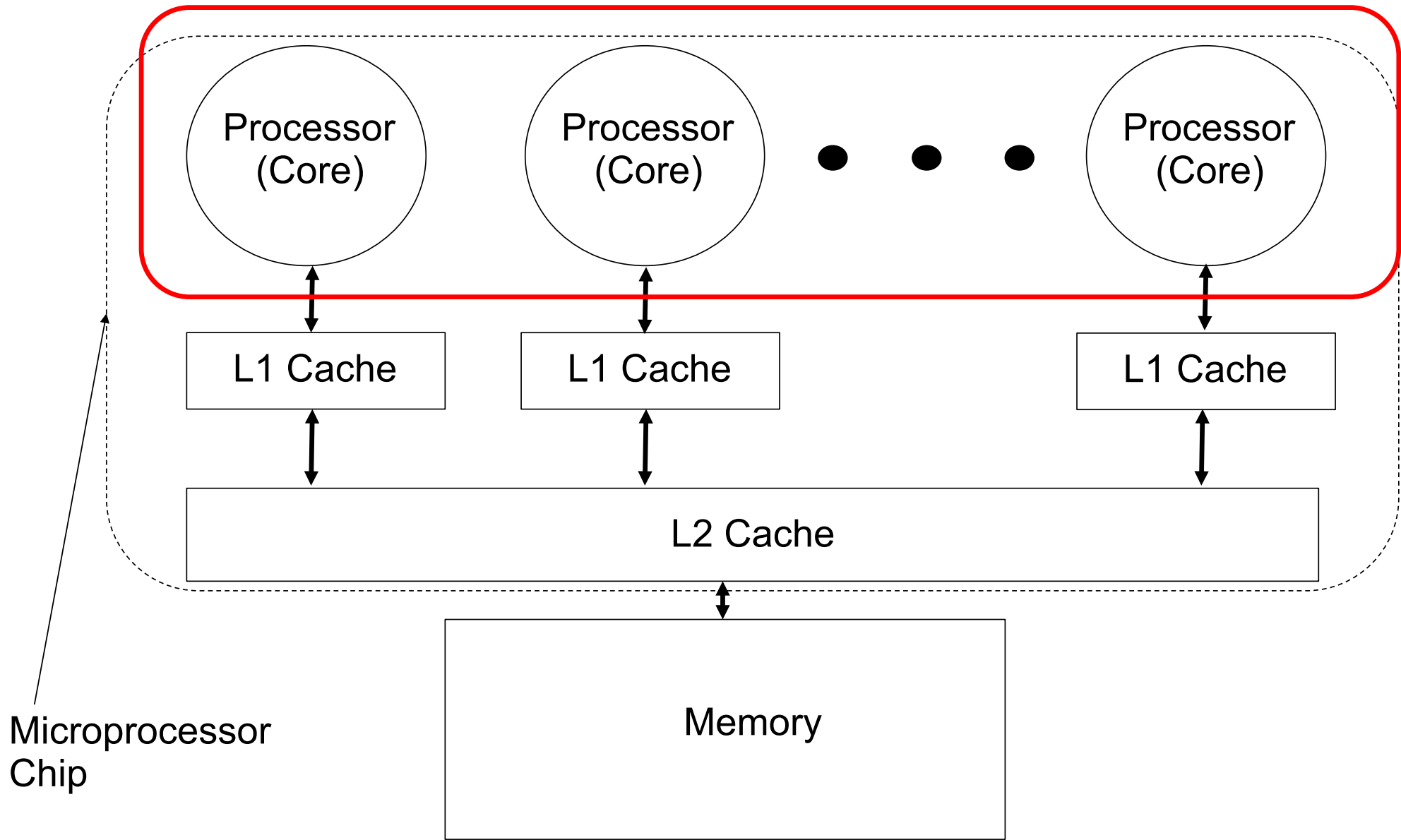
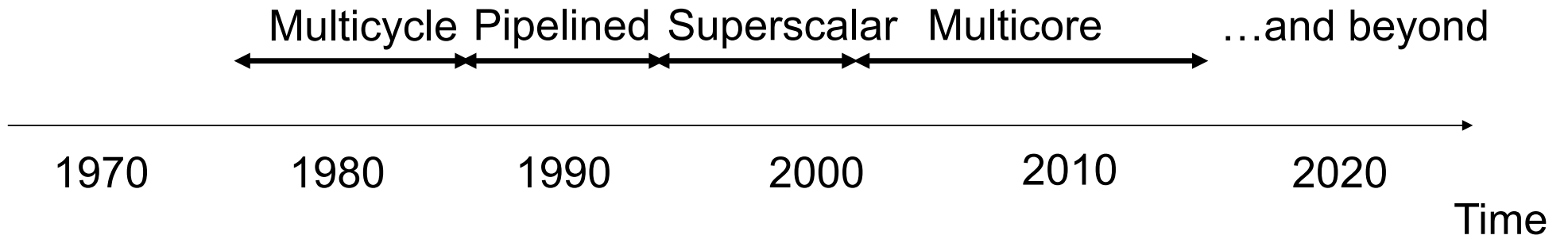


# The Evolution of Microprocessors

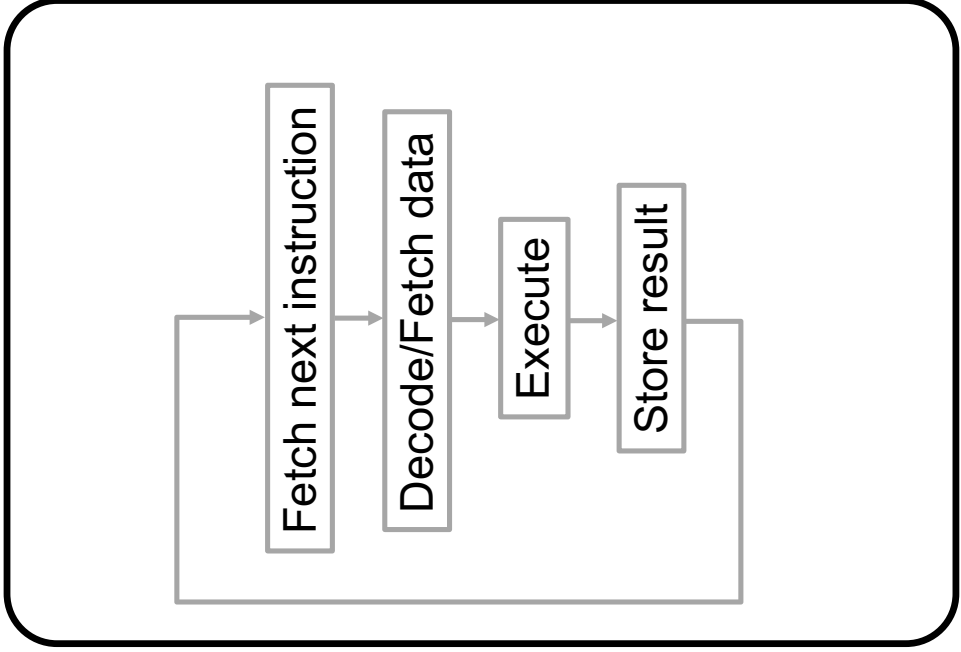
Per Stenström



# Evolution of Microprocessors



# Multicycle Computers

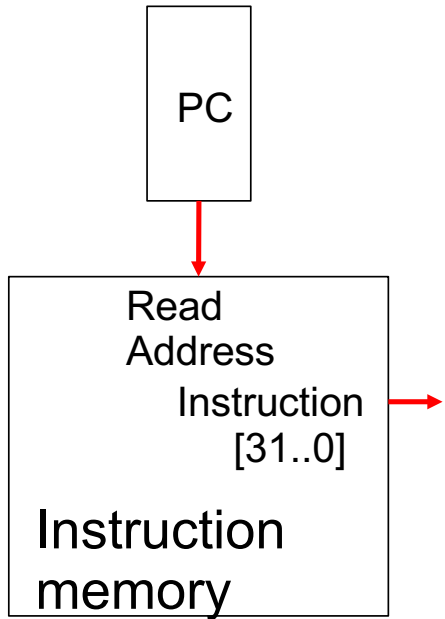


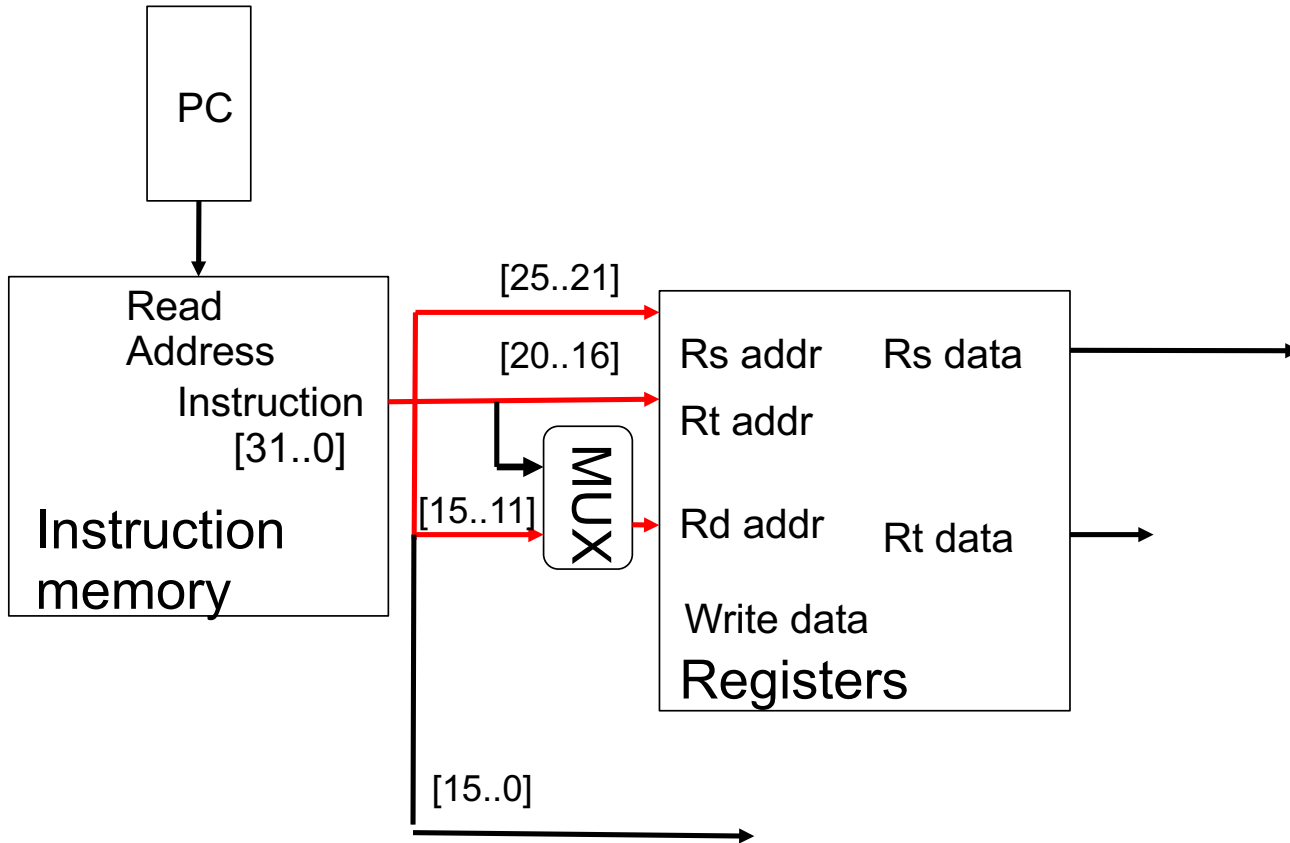
<b>Opcode</b>	<b>Assembly code</b>	<b>Meaning</b>	<b>Comments</b>
ADD,SUB,	ADD R1,R2,R3	$R1=(R2) + (R3)$	Integer operations
ADDI, SUBI	ADDI R1,R2,#3	$R1=(R2)+3$	Immediate
AND, OR, XOR	AND R1,R2,R3	$R1=(R2).AND.(R3)$	bit-wise logical AND,OR, XOR
SLT	SLT R1,R2,R3	If $(R2)<(R3)$ then $R1=1$ else $R1=0$	Test R2, R3 outcome in R1
ADD.D, SUB.D, MUL.D, DIV.D	ADD.D F1,F2,F3	$F1=(F2)+(F3)$	Floating-point operations

<b>Opcode</b>	<b>Assembly code</b>	<b>Meaning</b>	<b>Comments</b>
LB,LH,LW,LD	LW R1,#20(R2)	R1=MEM[20+(R2)]	For bytes, half-words, words and double words
SB,SH,SW,SD	SW R1,#20(R2)	MEM[20+(R2)]=R1	
L.S, L.D	L.S F0,#20(R2)	F0=MEM[20+(R2)]	Single/double float load
S.S, S.D	S.S F0,#20(R2)	MEM[20+(R2)]=F0	Single/double float store

<b>Opcode</b>	<b>Assembly code</b>	<b>Meaning</b>	<b>Comments</b>
BEQZ, BNEZ	BEQZ R1,Label	If (R1)=0 then PC=Label	Conditional branch-equal 0/not equal 0
BEQ, BNE	BEQ R1,R2,Label	If (R1)=(R2) PC=Label	Conditional branch- equal/not equal
J	J Target	PC=Target	Target is an immediate field
JR	JR R1	PC=(R1)	Target is in register
JAL	JAL Target	R31 = PC + 4; PC = Target	Jump to target and save return address in R31



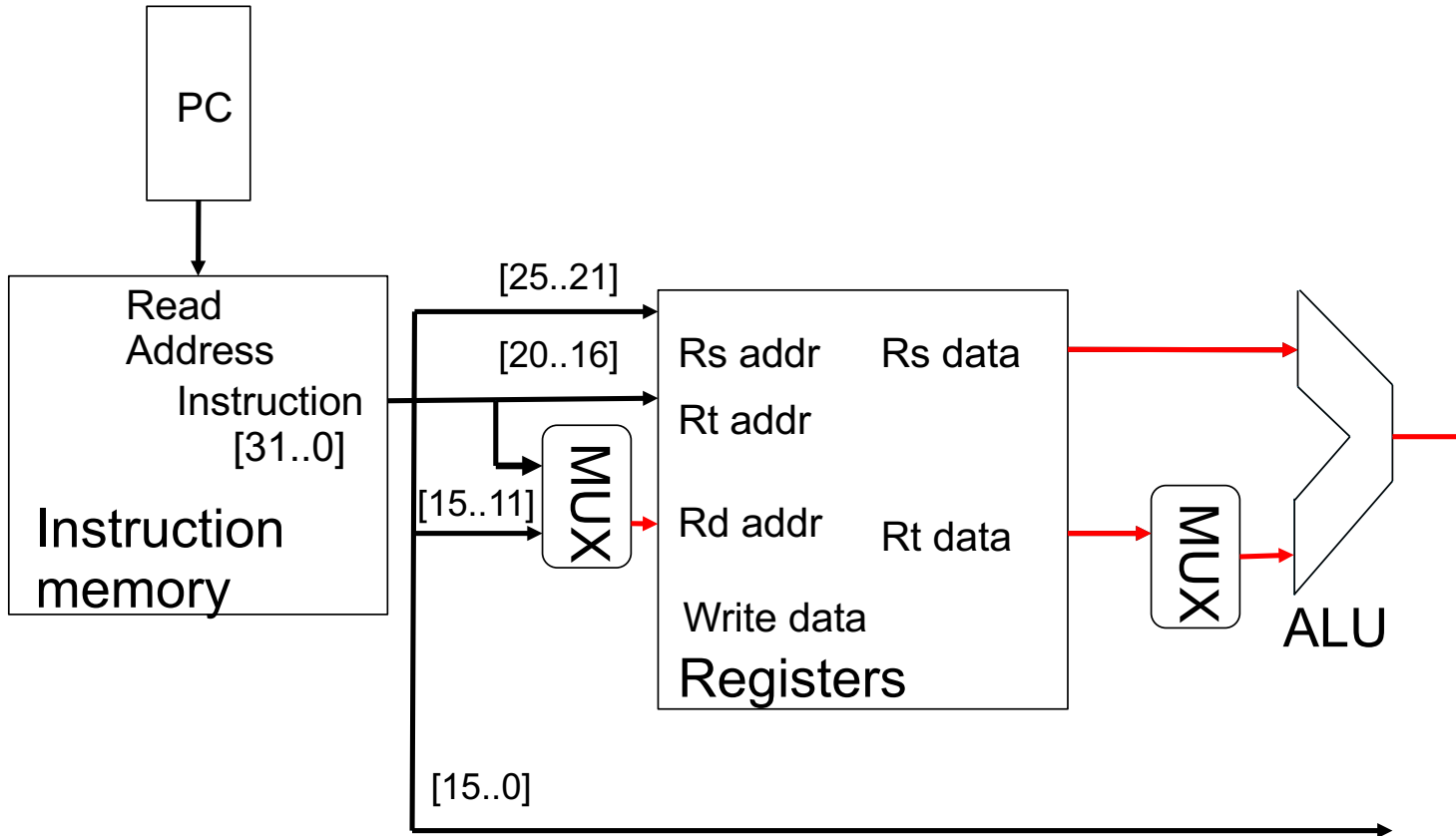




ALU

Memory transfer

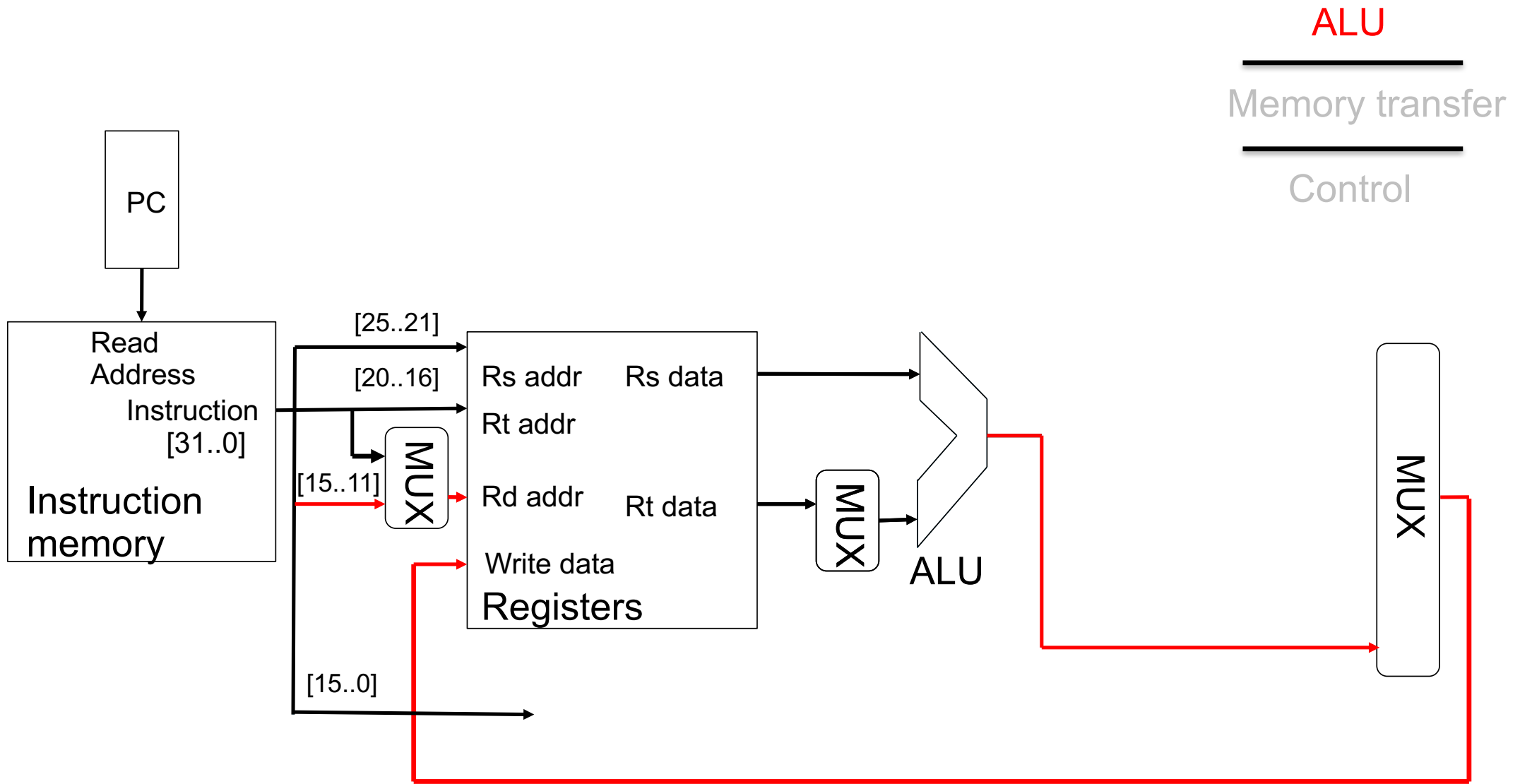
Control

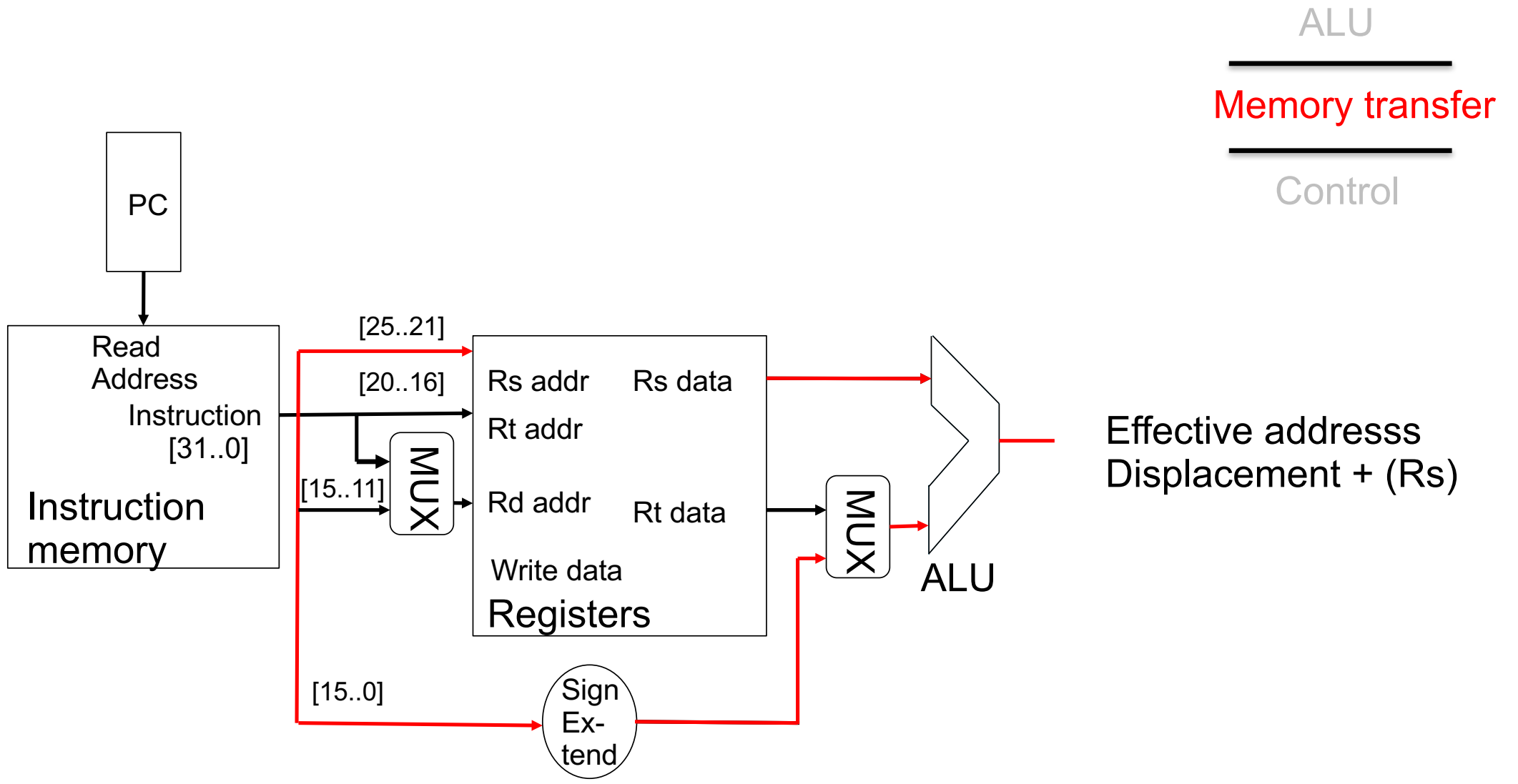


ALU

Memory transfer

Control



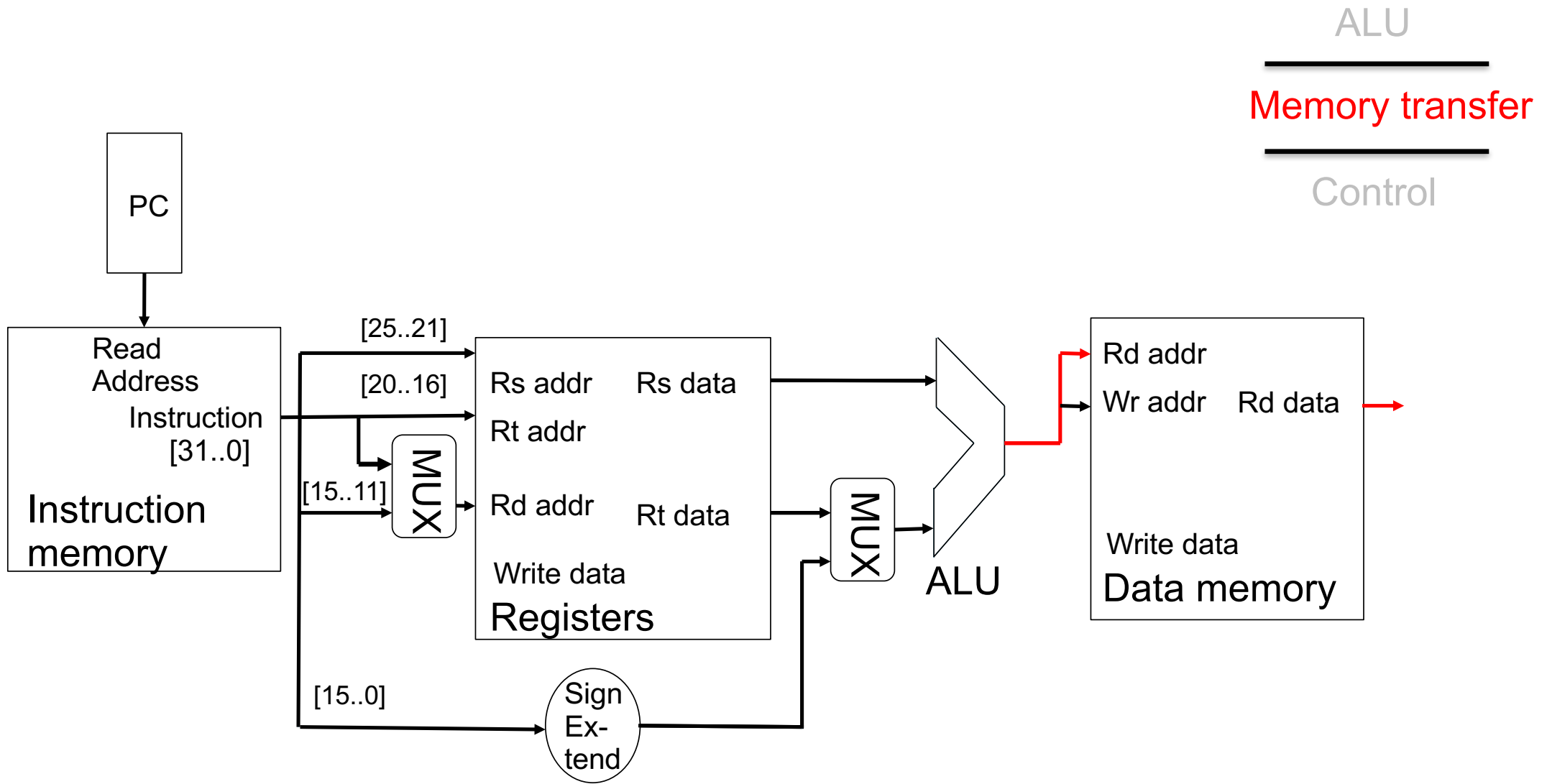


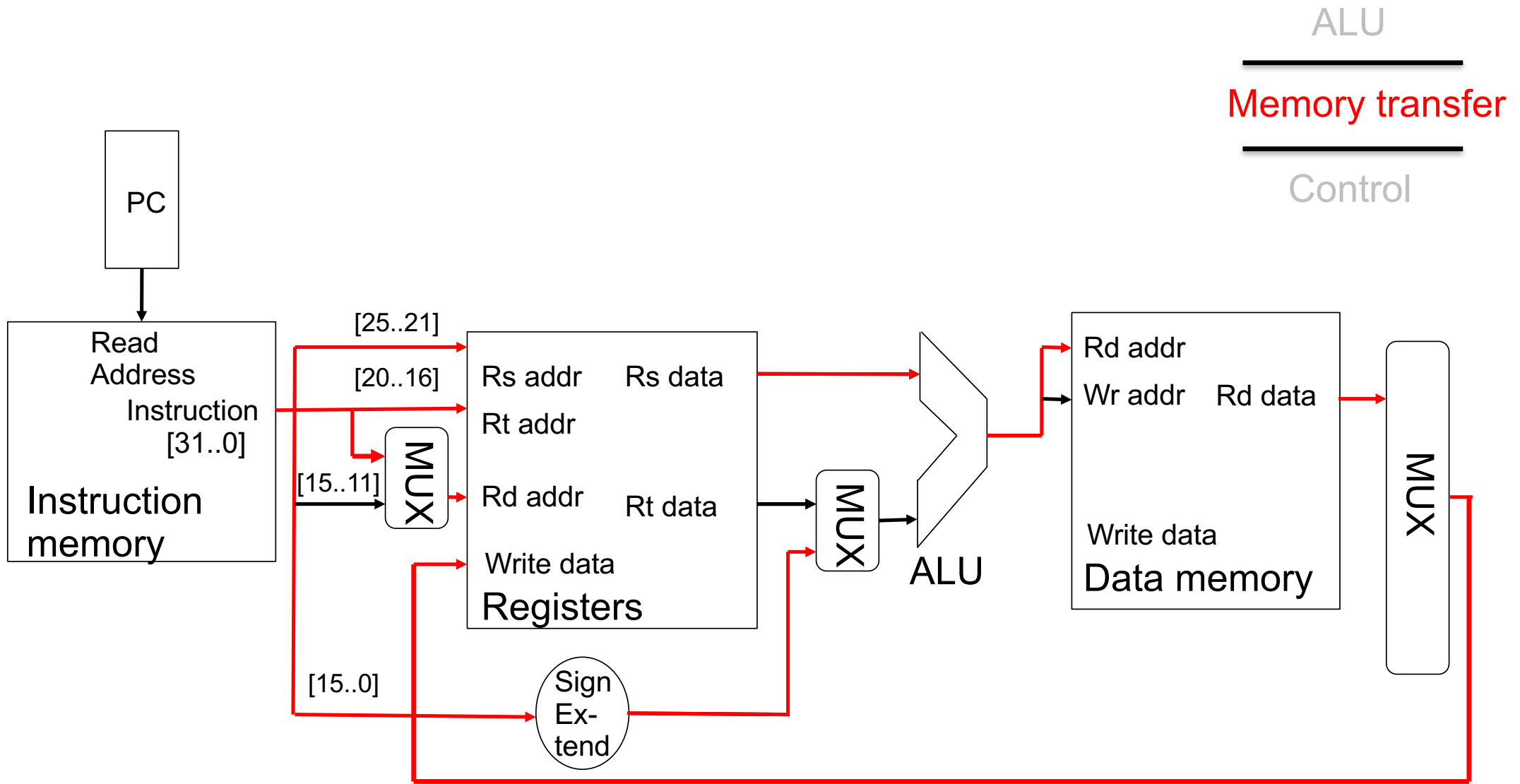
ALU

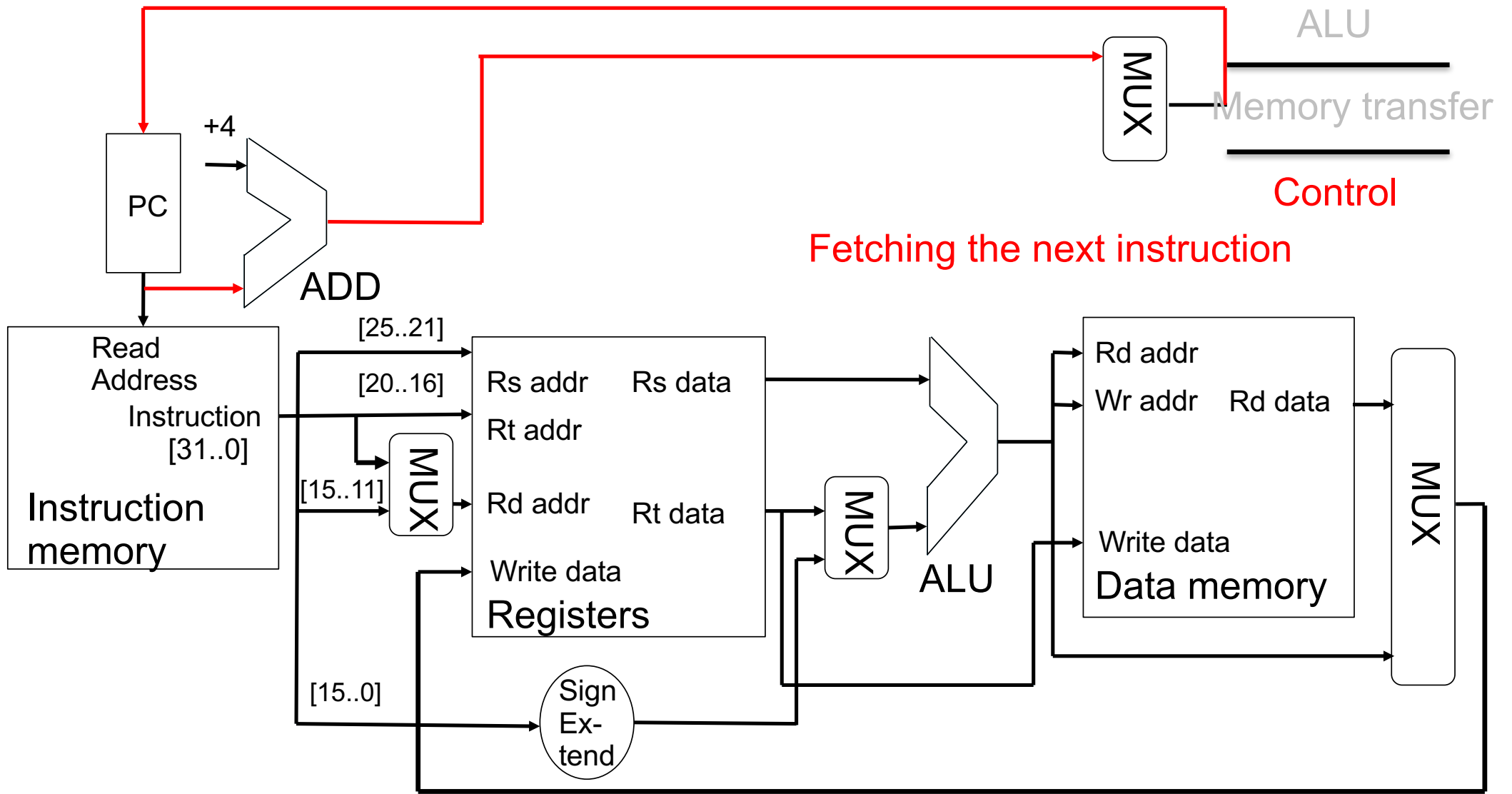
Memory transfer

Control

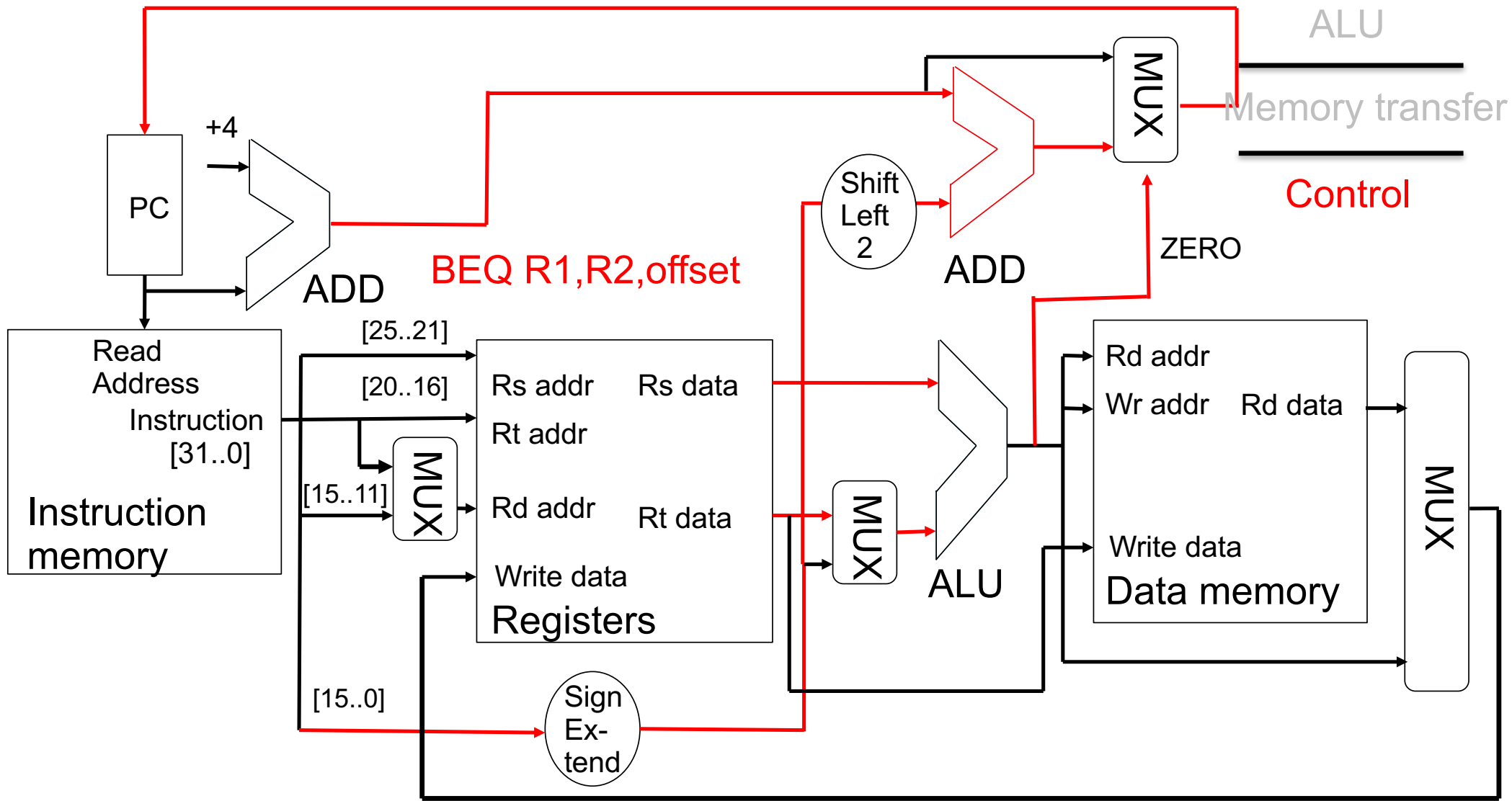
Effective address  
Displacement + (Rs)

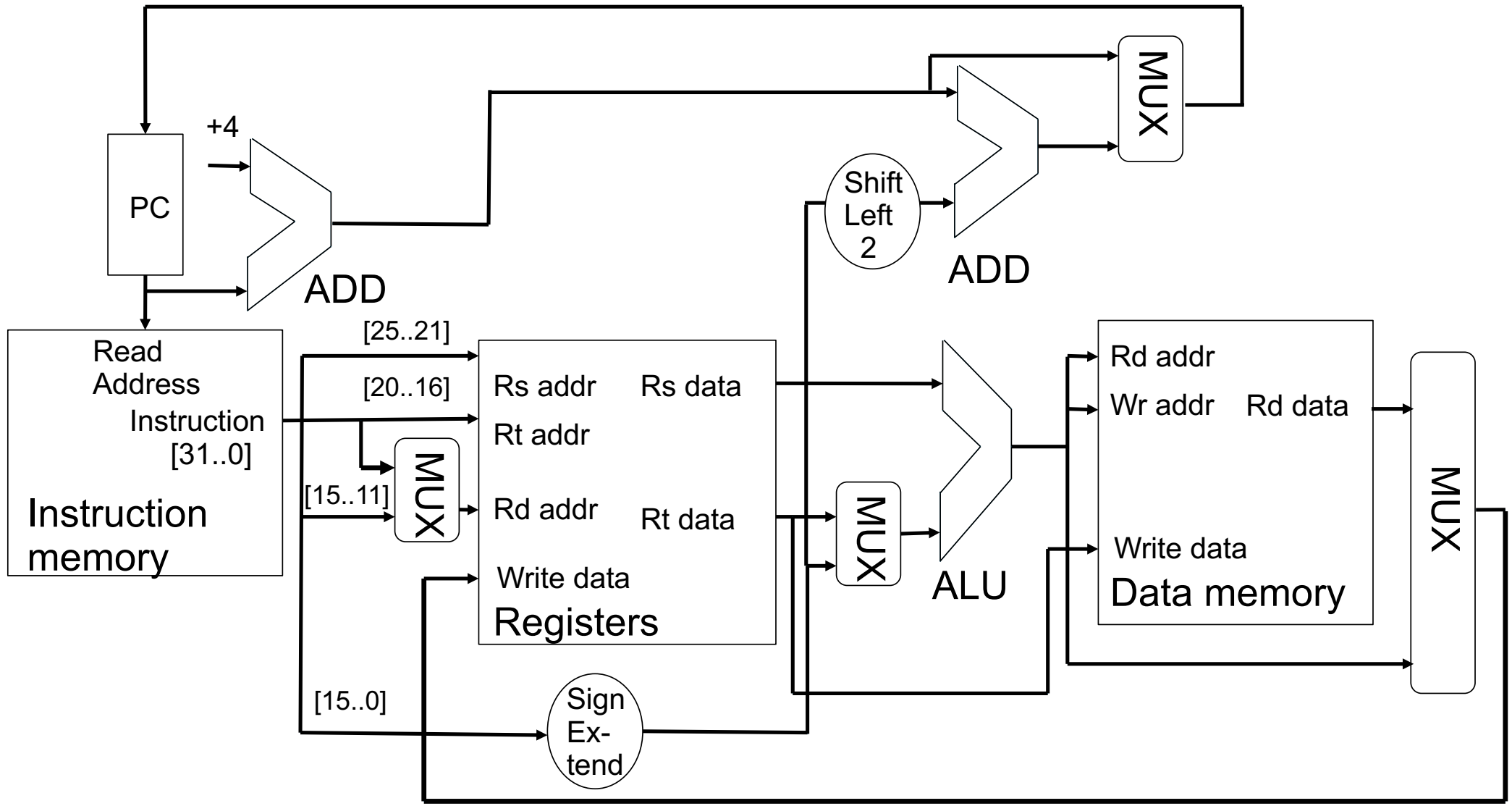


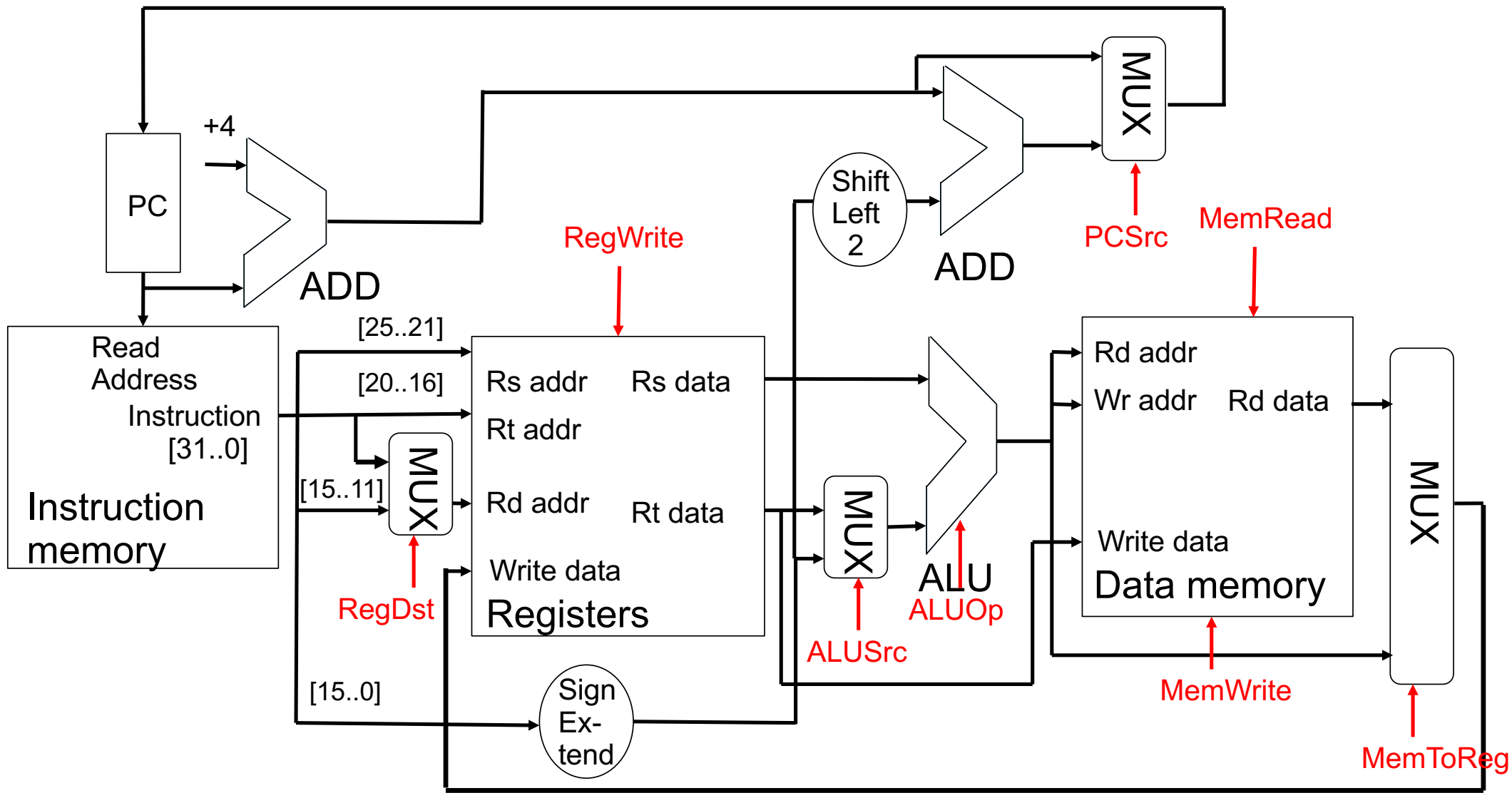


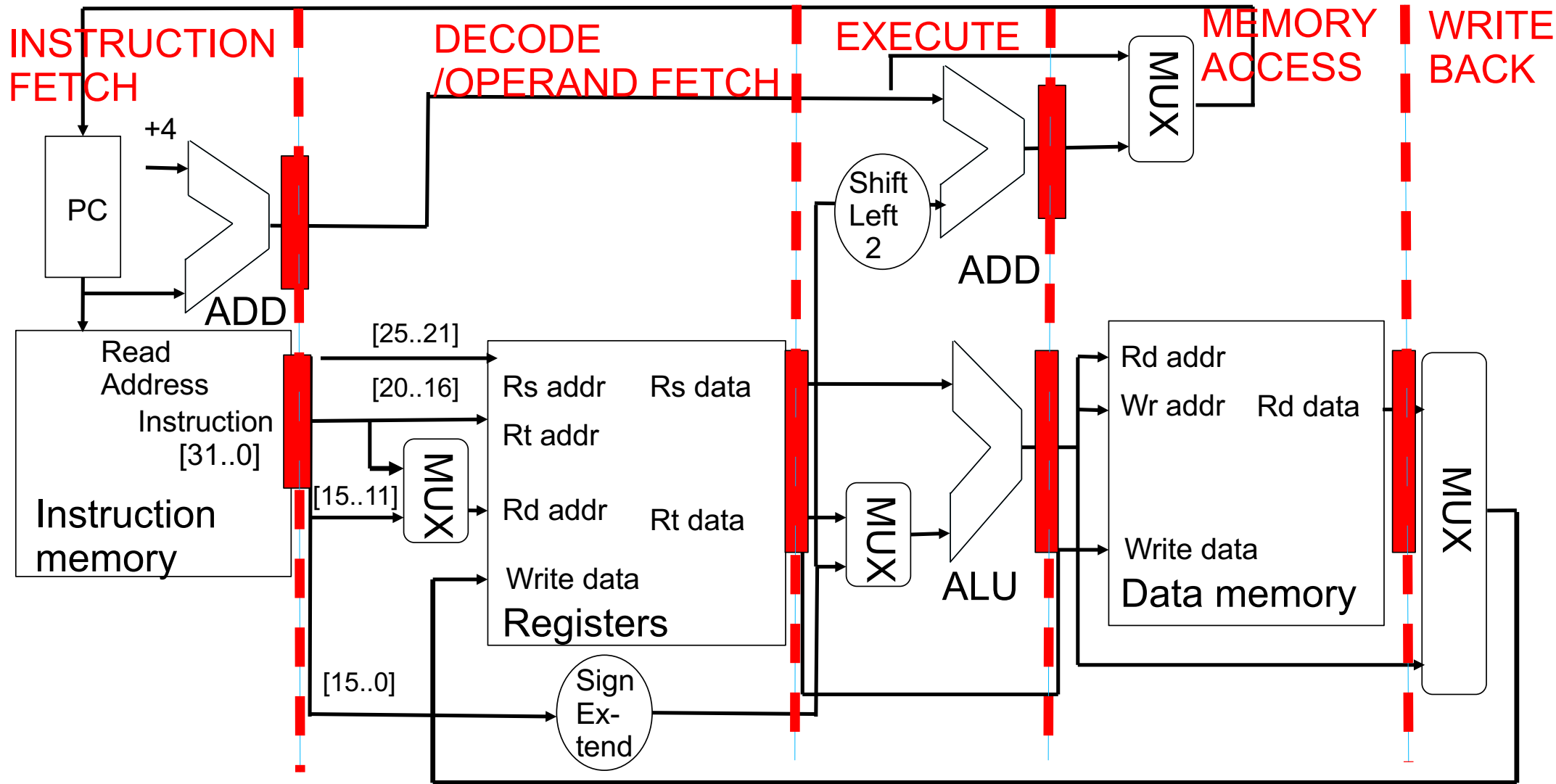






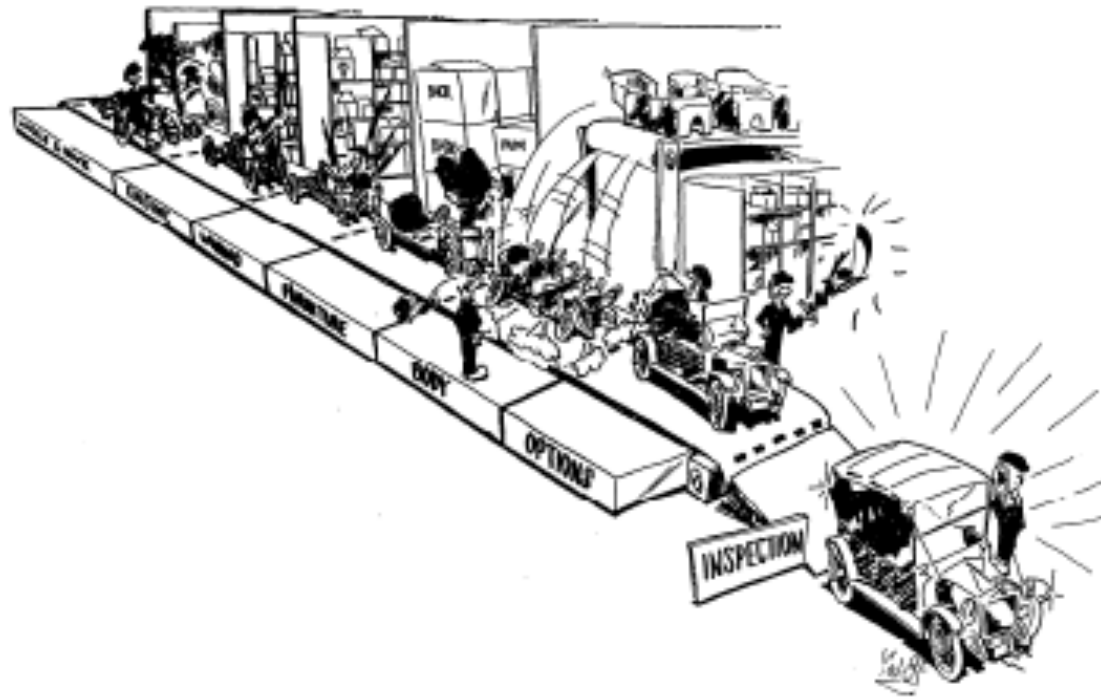


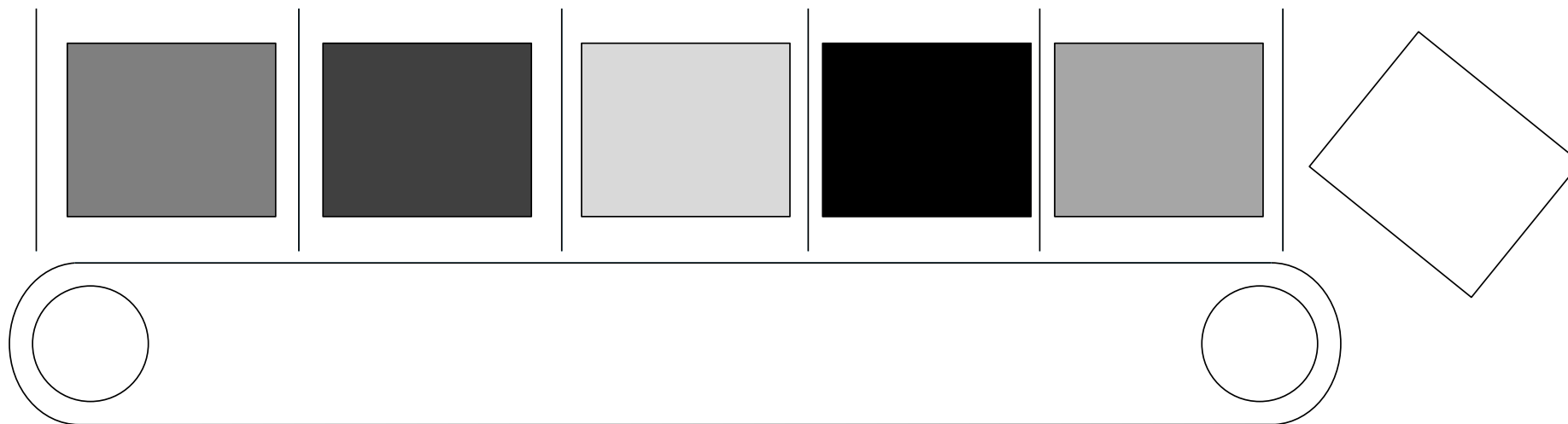


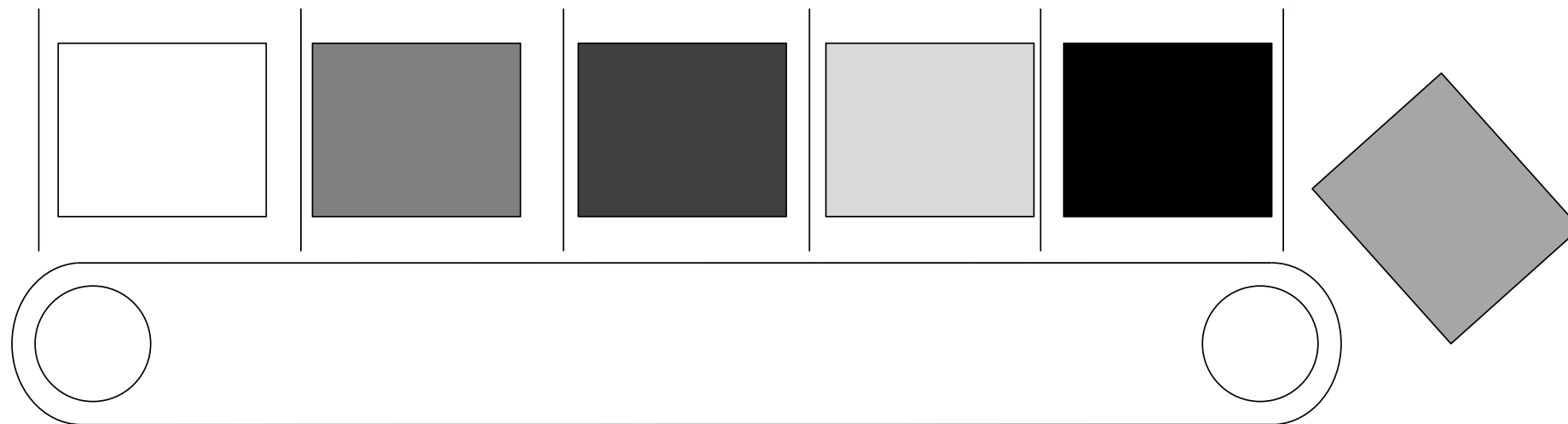


# Pipelined "Single-Cycle" Computer

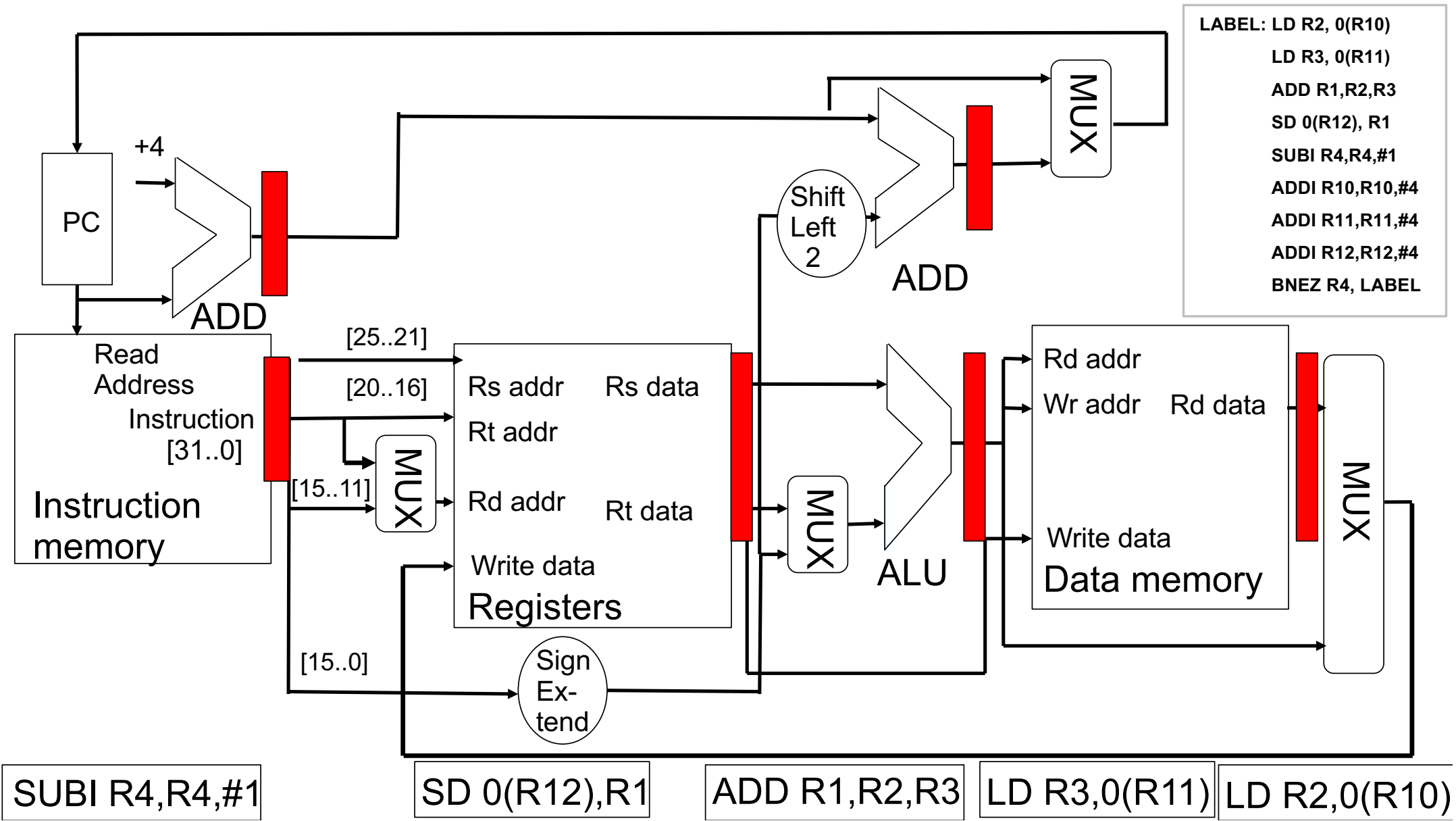
# The Conveyor Belt







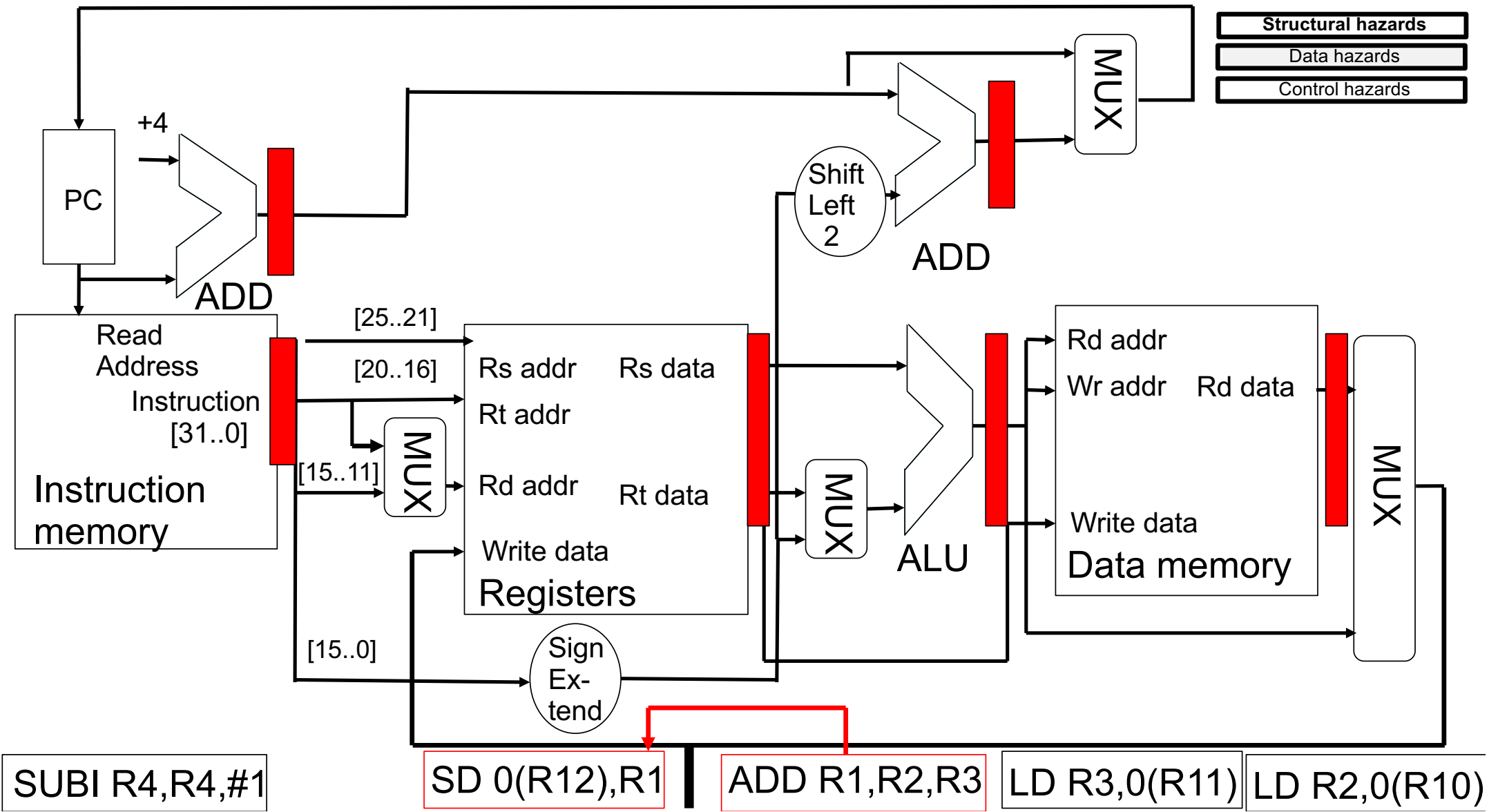


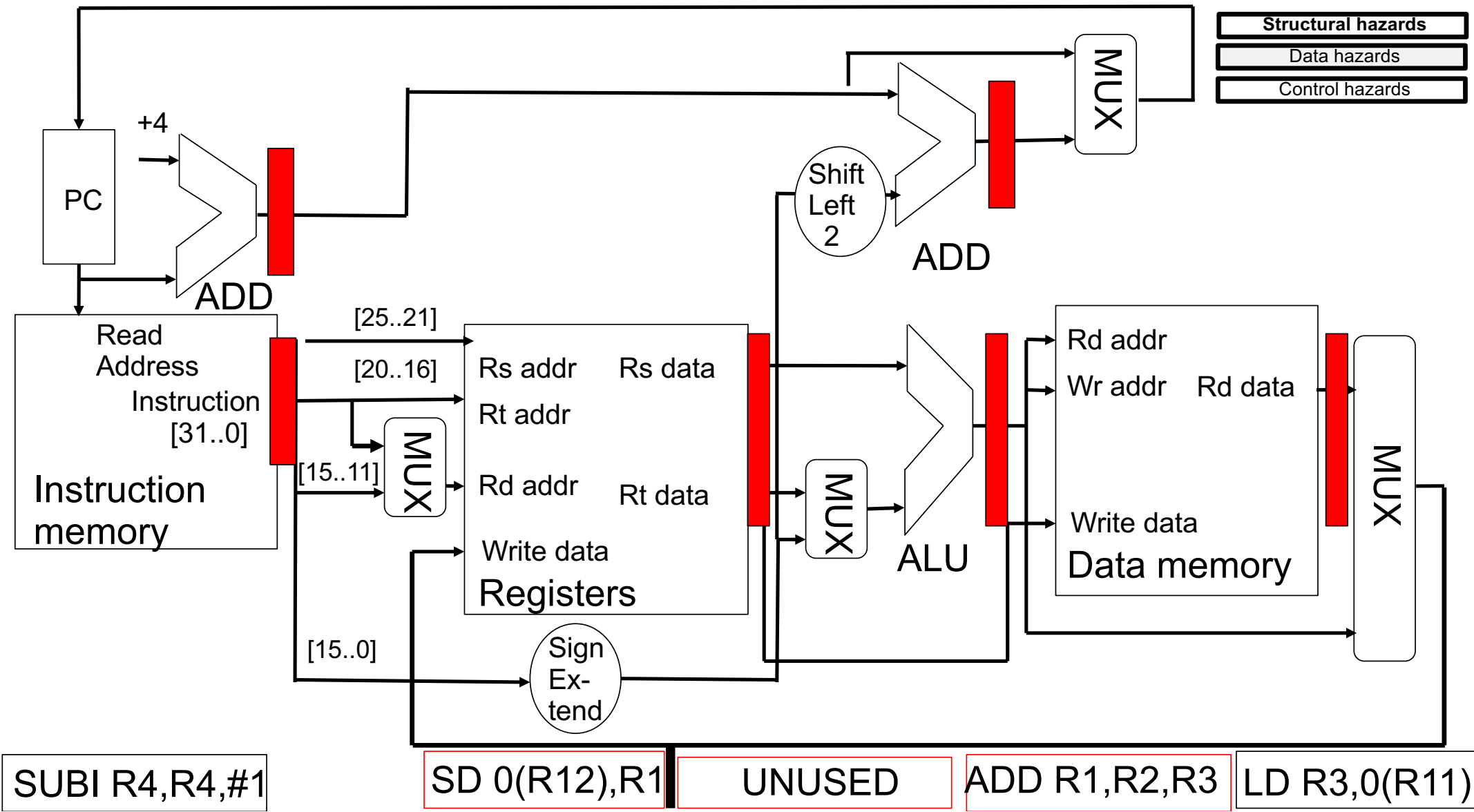


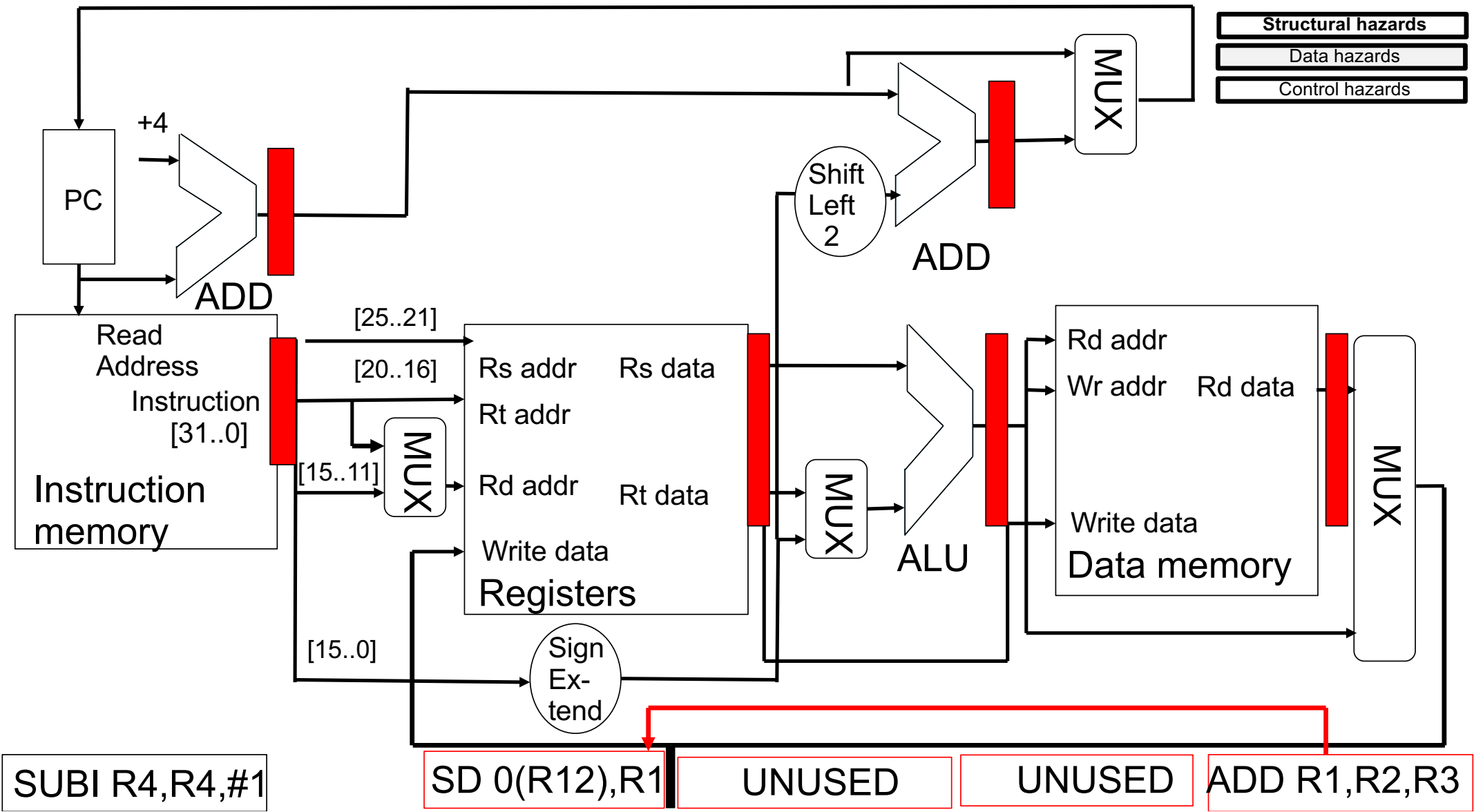
```

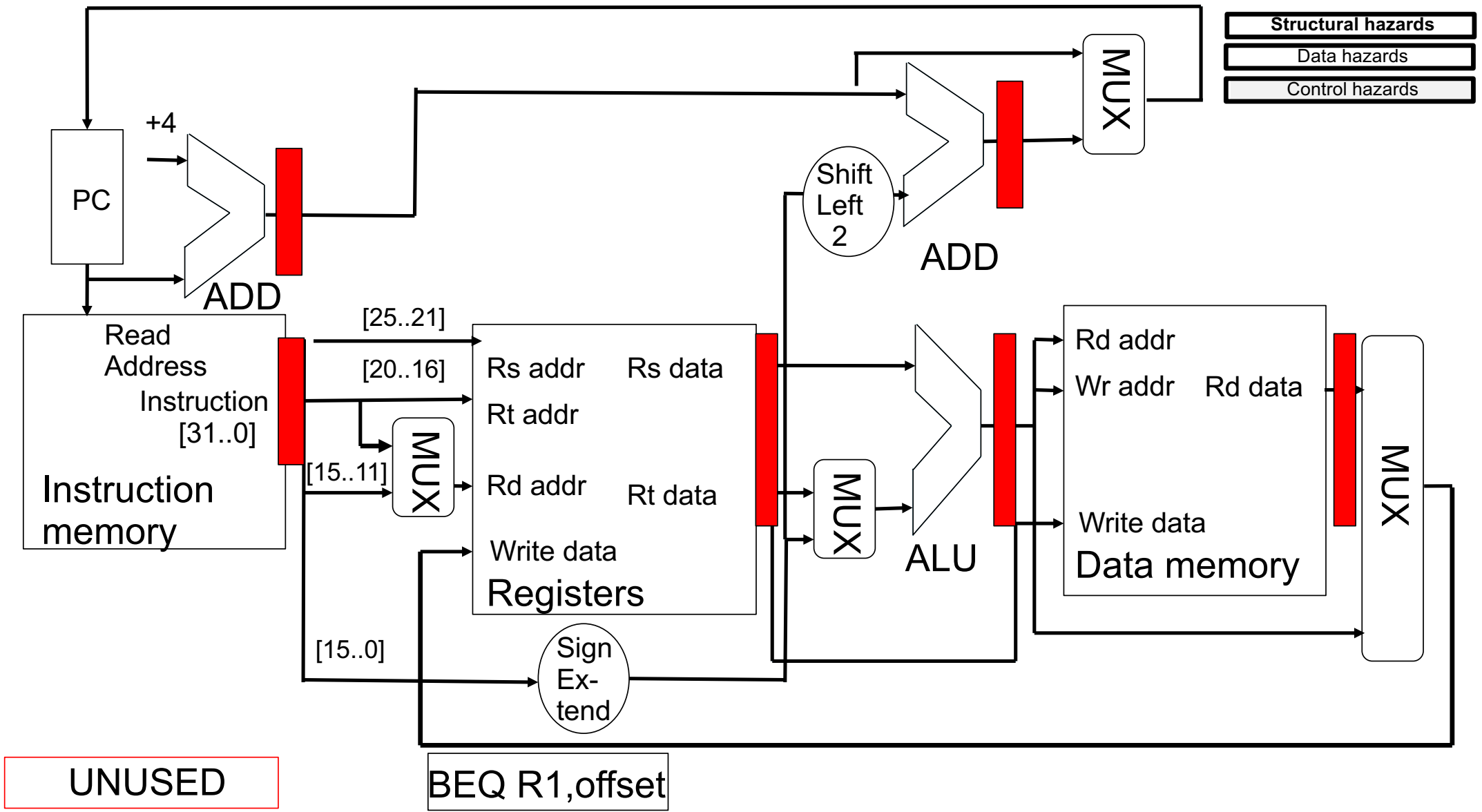
LABEL: LD R2, 0(R10)
LD R3, 0(R11)
ADD R1,R2,R3
SD 0(R12), R1
SUBI R4,R4,#1
ADDI R10,R10,#4
ADDI R11,R11,#4
ADDI R12,R12,#4
BNEZ R4, LABEL

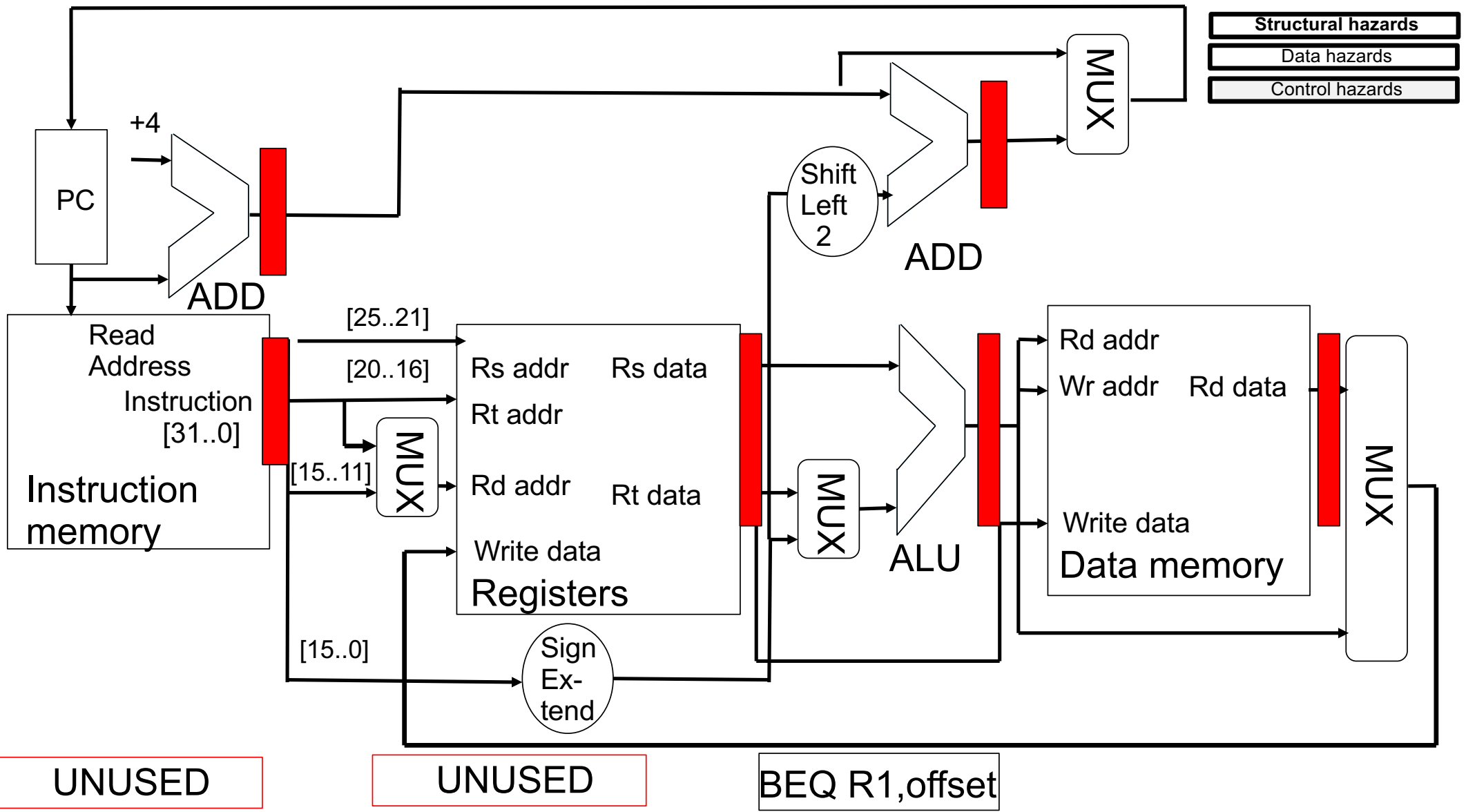
```









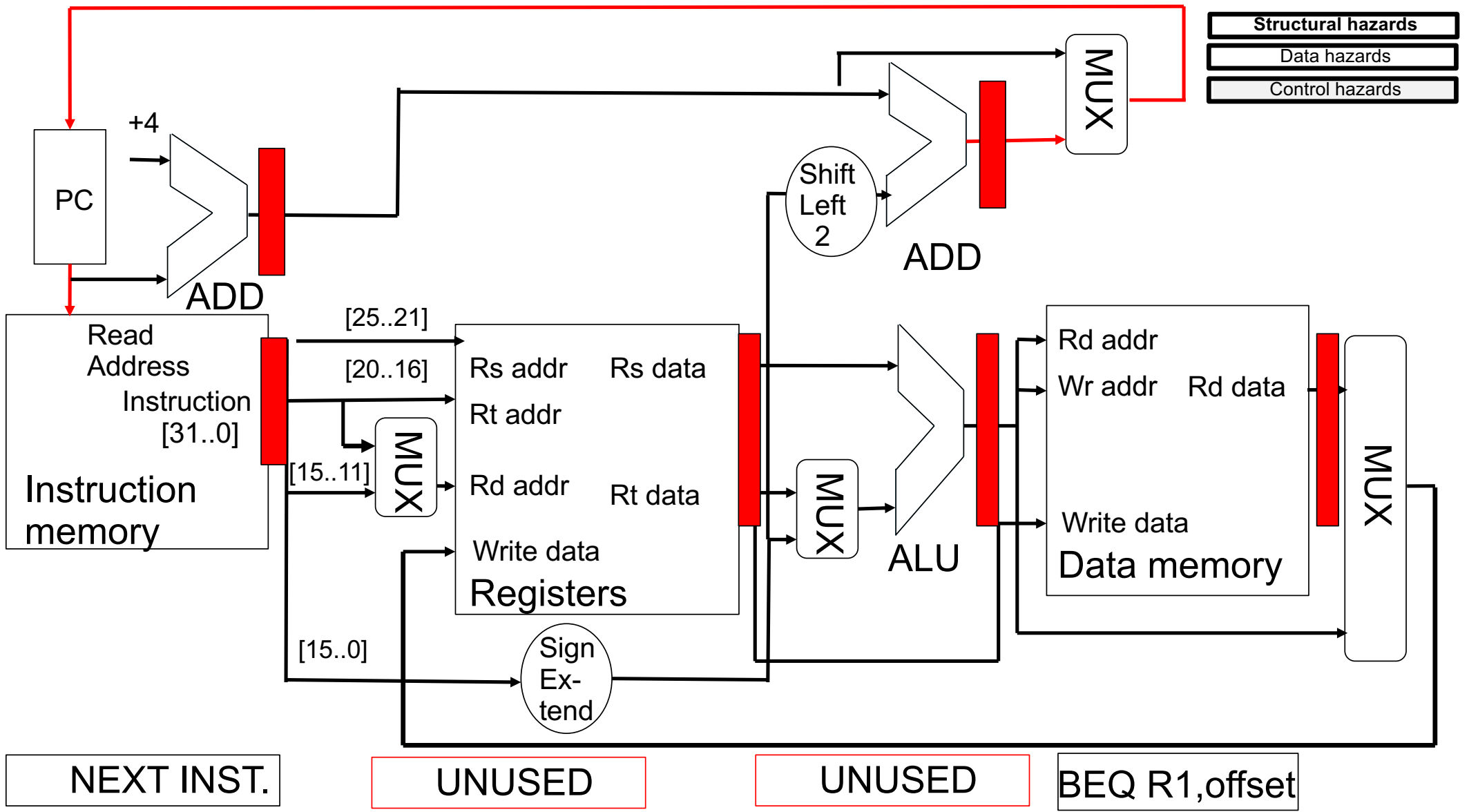


- Structural hazards**
- Data hazards**
- Control hazards**

UNUSED

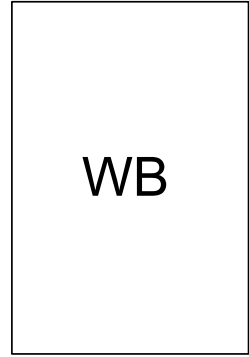
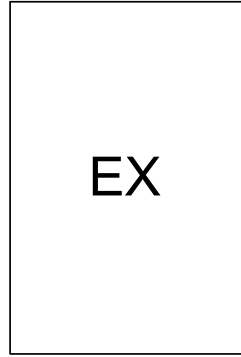
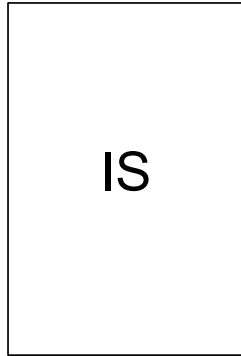
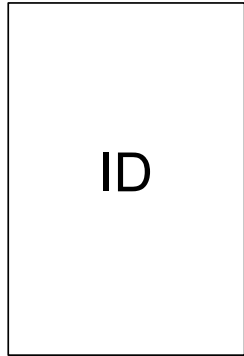
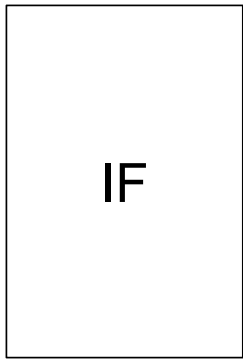
UNUSED

BEQ R1,offset



# Superscalar Microprocessors





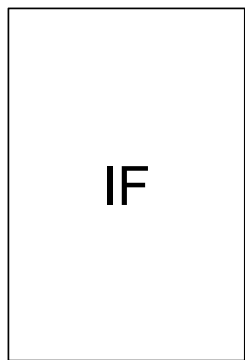
Instruction  
Fetch

Instruction  
Decode/  
Dispatch

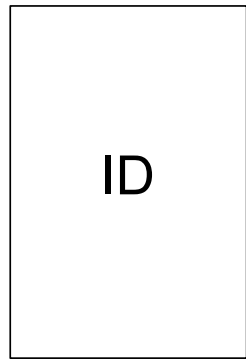
Instruction  
Issue/Read  
Operands

Execute

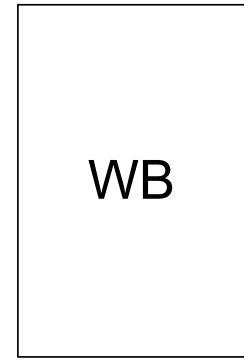
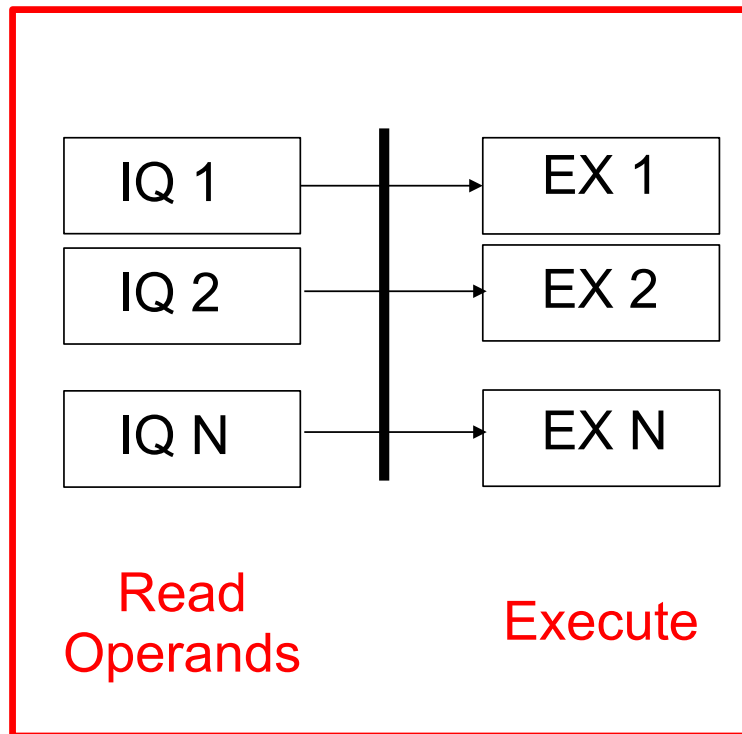
Write  
Back



Fetch



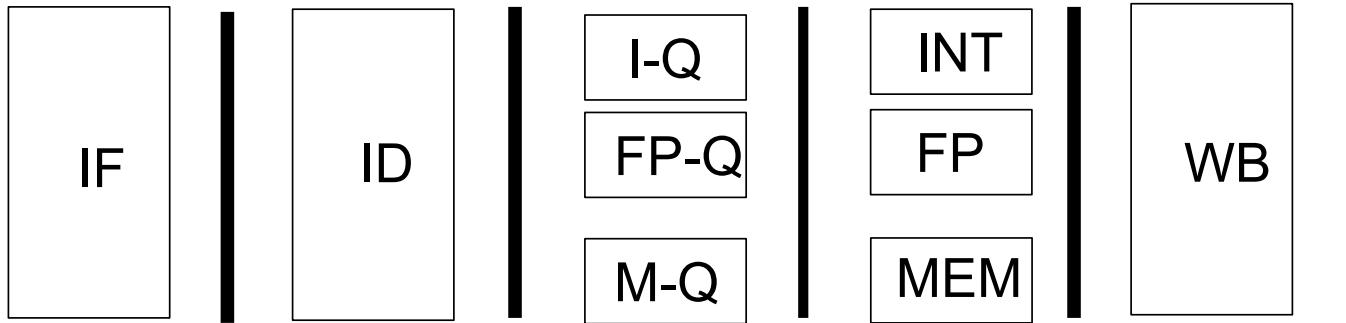
Decode/  
Dispatch



Write  
Back



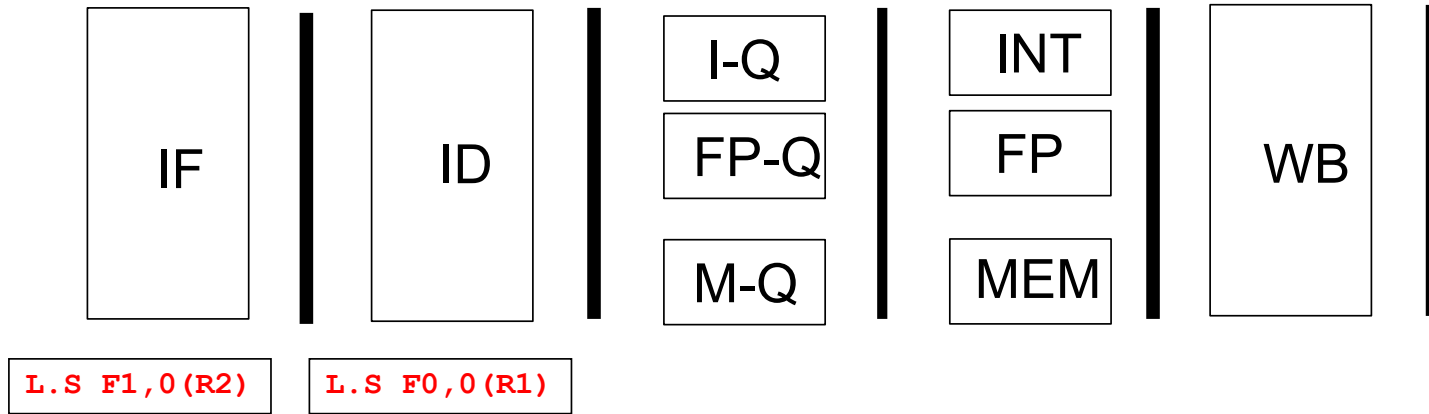
Cycle: 1



L.S F0,0(R1)

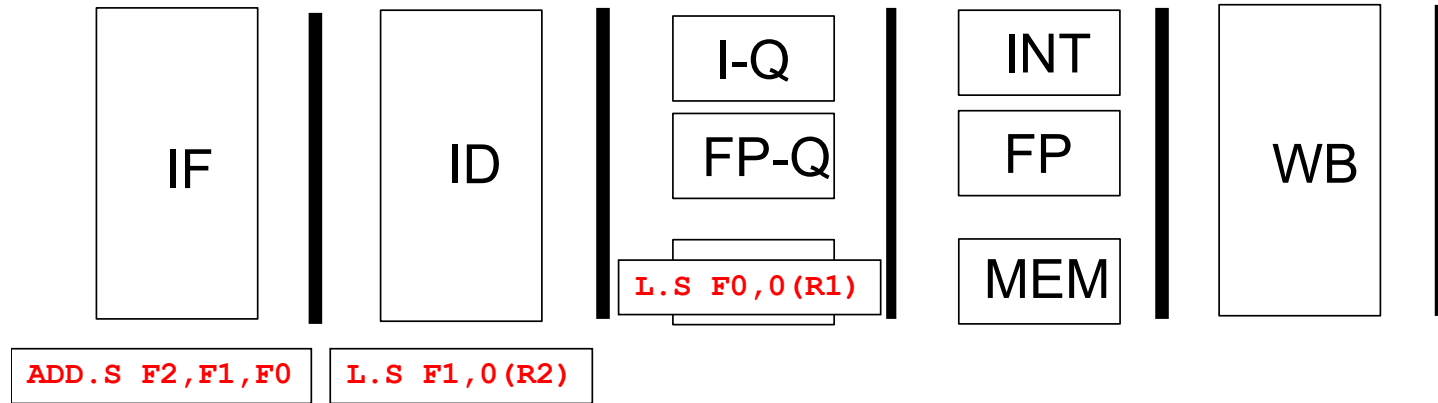
Loop → L.S F0,0(R1)  
L.S F1,0(R2)  
ADD.S F2,F1,F0  
S.S F2,0(R1)  
ADDI R1,R1,#4  
ADDI R2,R2,#4  
SUBI R3,R3,#1  
BNEZ R3,Loop

Cycle: 2



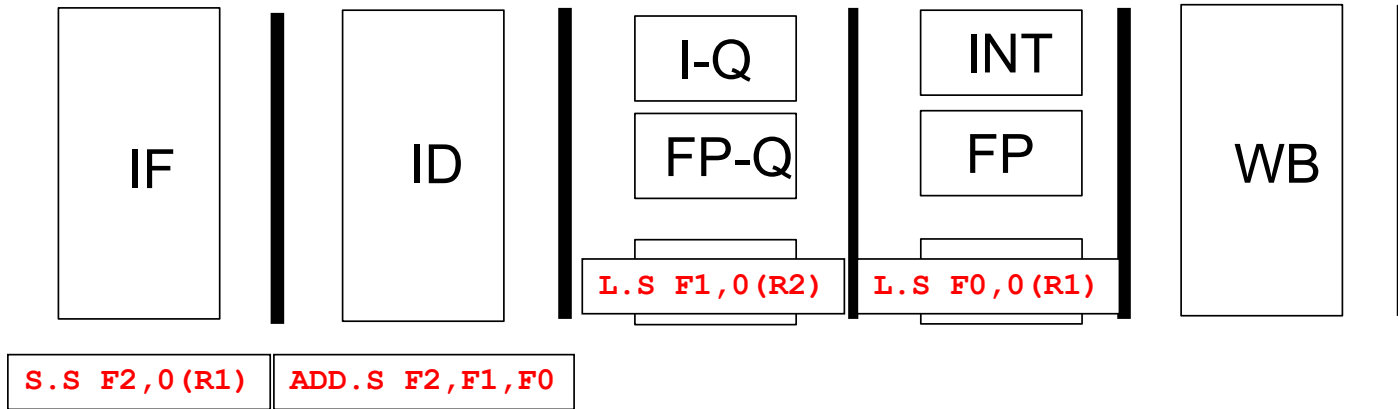
```
Loop    L.S F0,0(R1)
         L.S F1,0(R2)
         ADD.S F2,F1,F0
         S.S F2,0(R1)
         ADDI R1,R1,#4
         ADDI R2,R2,#4
         SUBI R3,R3,#1
         BNEZ R3,Loop
```

Cycle: 3



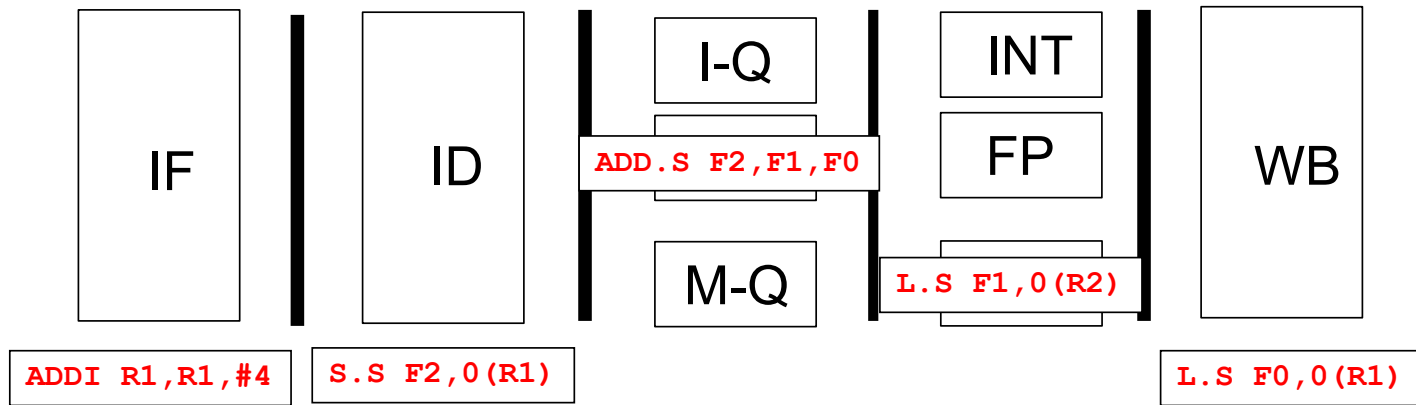
```
Loop      L.S F0,0(R1)
          L.S F1,0(R2)
          ➔ ADD.S F2,F1,F0
          S.S F2,0(R1)
          ADDI R1,R1,#4
          ADDI R2,R2,#4
          SUBI R3,R3,#1
          BNEZ R3,Loop
```

Cycle: 4



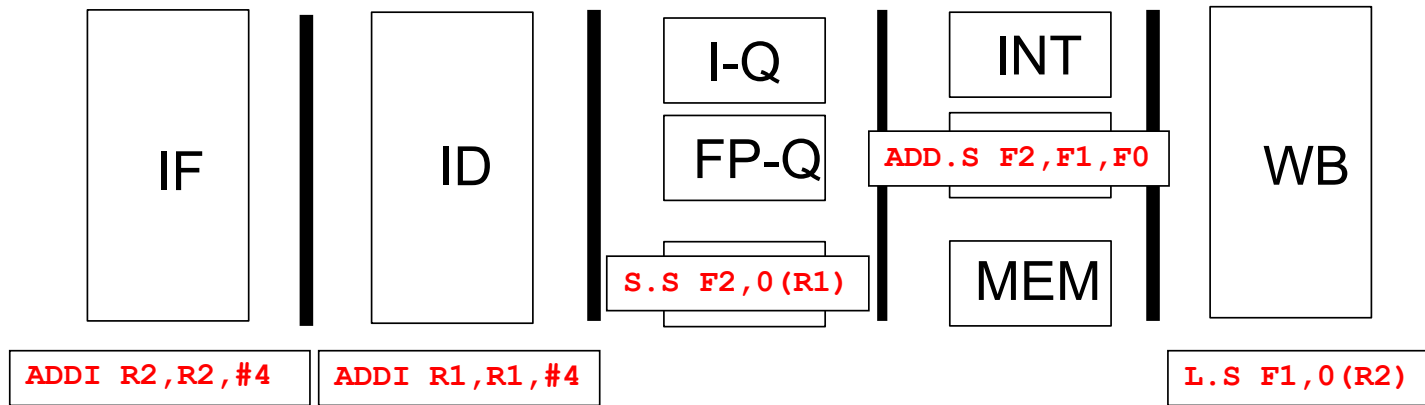
Loop    L.S F0,0(R1)  
          L.S F1,0(R2)  
          ADD.S F2,F1,F0  
          S.S F2,0(R1)  
          ADDI R1,R1,#4  
          ADDI R2,R2,#4  
          SUBI R3,R3,#1  
          BNEZ R3,Loop

Cycle: 5



```
Loop    L.S F0, 0(R1)
        L.S F1, 0(R2)
        ADD.S F2, F1, F0
        S.S F2, 0(R1)
        ADDI R1, R1, #4
        ADDI R2, R2, #4
        SUBI R3, R3, #1
        BNEZ R3, Loop
```

Cycle: 6

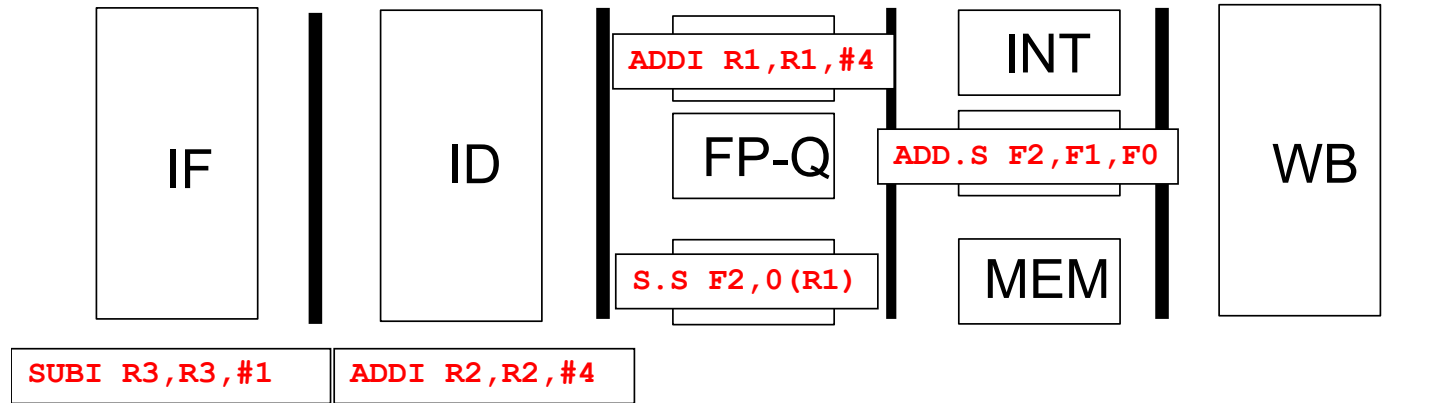


Loop    L.S F0, 0(R1)  
          L.S F1, 0(R2)  
          ADD.S F2, F1, F0  
          S.S F2, 0(R1)  
          ADDI R1, R1, #4  
          ADDI R2, R2, #4  
          SUBI R3, R3, #1  
          BNEZ R3, Loop



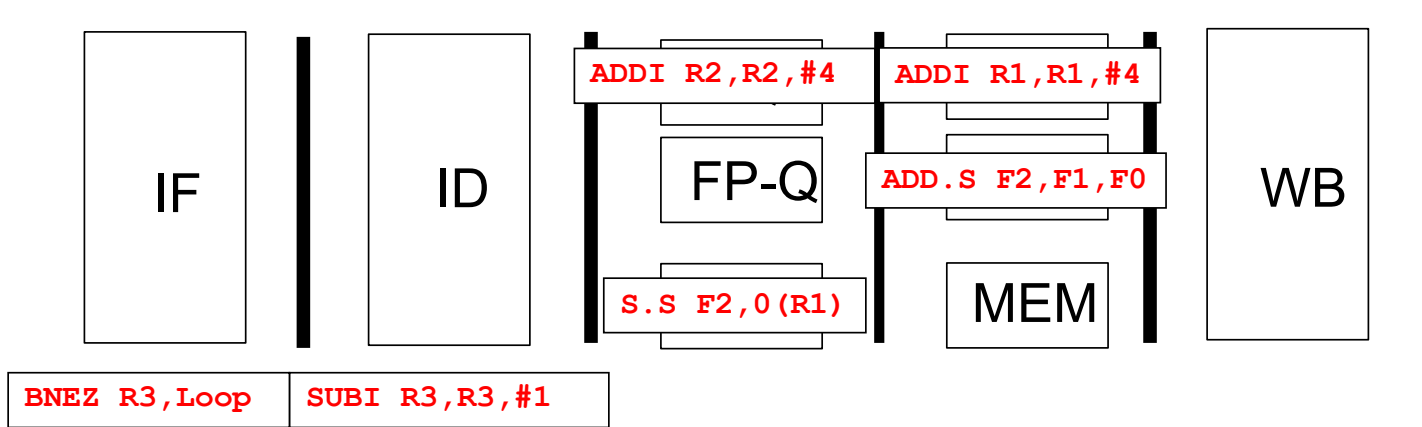


Cycle: 7



Loop    L.S F0, 0(R1)  
         L.S F1, 0(R2)  
         ADD.S F2, F1, F0  
         S.S F2, 0(R1)  
         ADDI R1, R1, #4  
         ADDI R2, R2, #4  
         SUBI R3, R3, #1  
         BNEZ R3, Loop





Cycle: 8

The `ADDI R1,R1,#4` is being executed Out-of-program-order with respect to the `S.S F2,0(R1)`!

Loop

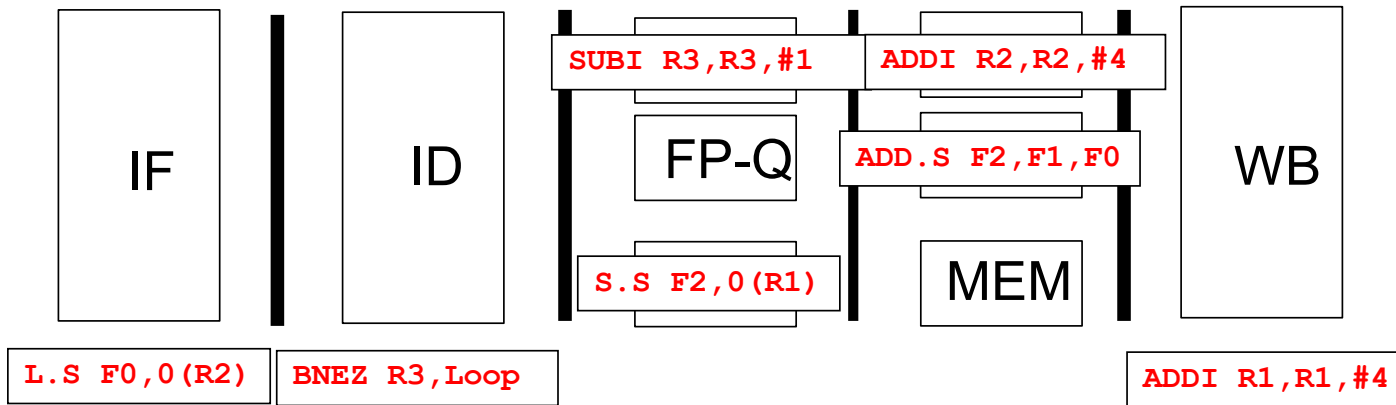
```

L.S F0,0(R1)
L.S F1,0(R2)
ADD.S F2,F1,F0
S.S F2,0(R1)
ADDI R1,R1,#4
ADDI R2,R2,#4
SUBI R3,R3,#1
BNEZ R3,Loop

```

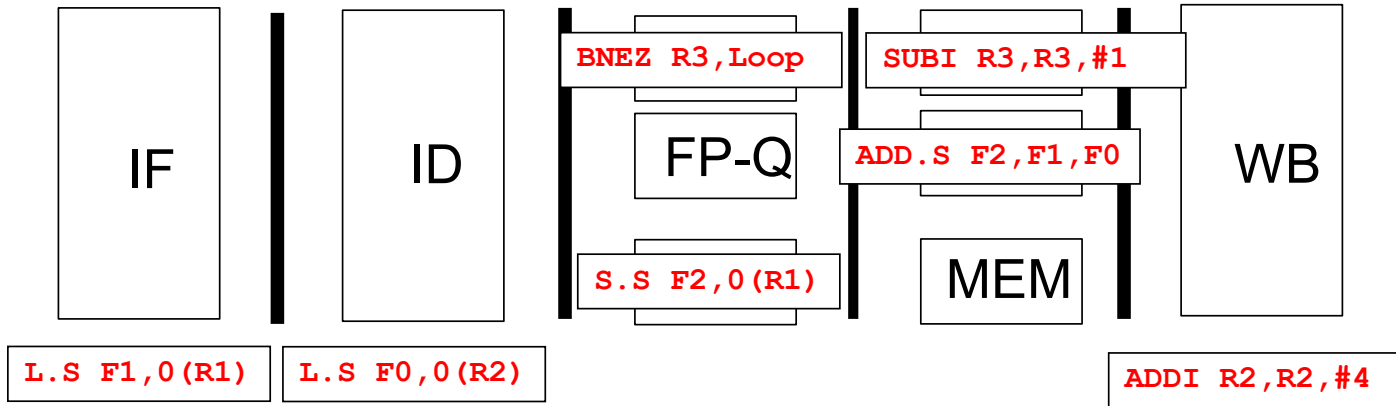
→

Cycle: 9



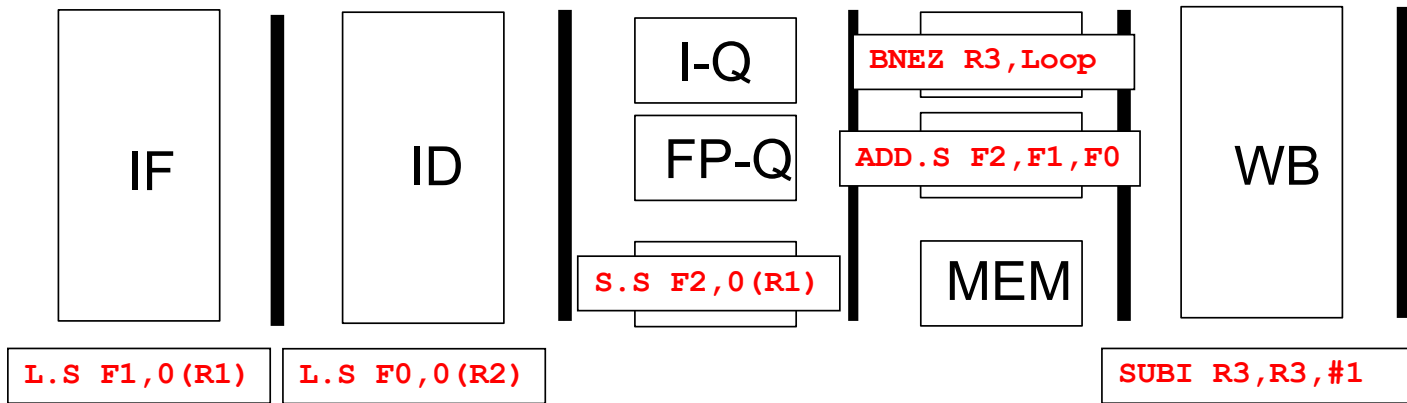
Loop → L.S F0,0(R1)  
L.S F1,0(R2)  
ADD.S F2,F1,F0  
S.S F2,0(R1)  
ADDI R1,R1,#4  
ADDI R2,R2,#4  
SUBI R3,R3,#1  
BNEZ R3,Loop

Cycle: 10



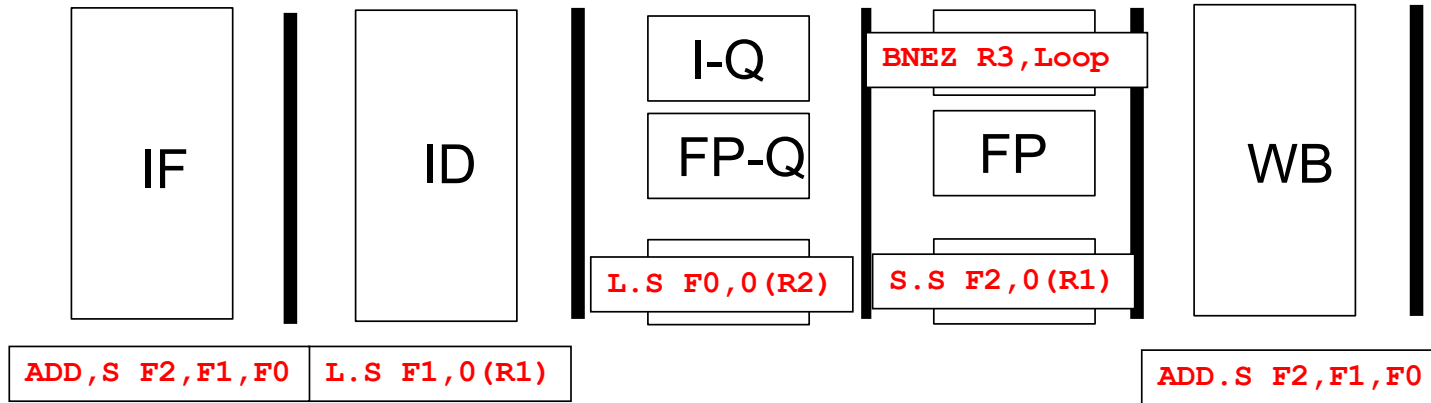
Loop → L.S F0,0(R1)  
L.S F1,0(R2)  
ADD.S F2,F1,F0  
S.S F2,0(R1)  
ADDI R1,R1,#4  
ADDI R2,R2,#4  
SUBI R3,R3,#1  
BNEZ R3,Loop

Cycle: 11



Loop → L.S F0,0(R1)  
L.S F1,0(R2)  
ADD.S F2,F1,F0  
S.S F2,0(R1)  
ADDI R1,R1,#4  
ADDI R2,R2,#4  
SUBI R3,R3,#1  
BNEZ R3,Loop

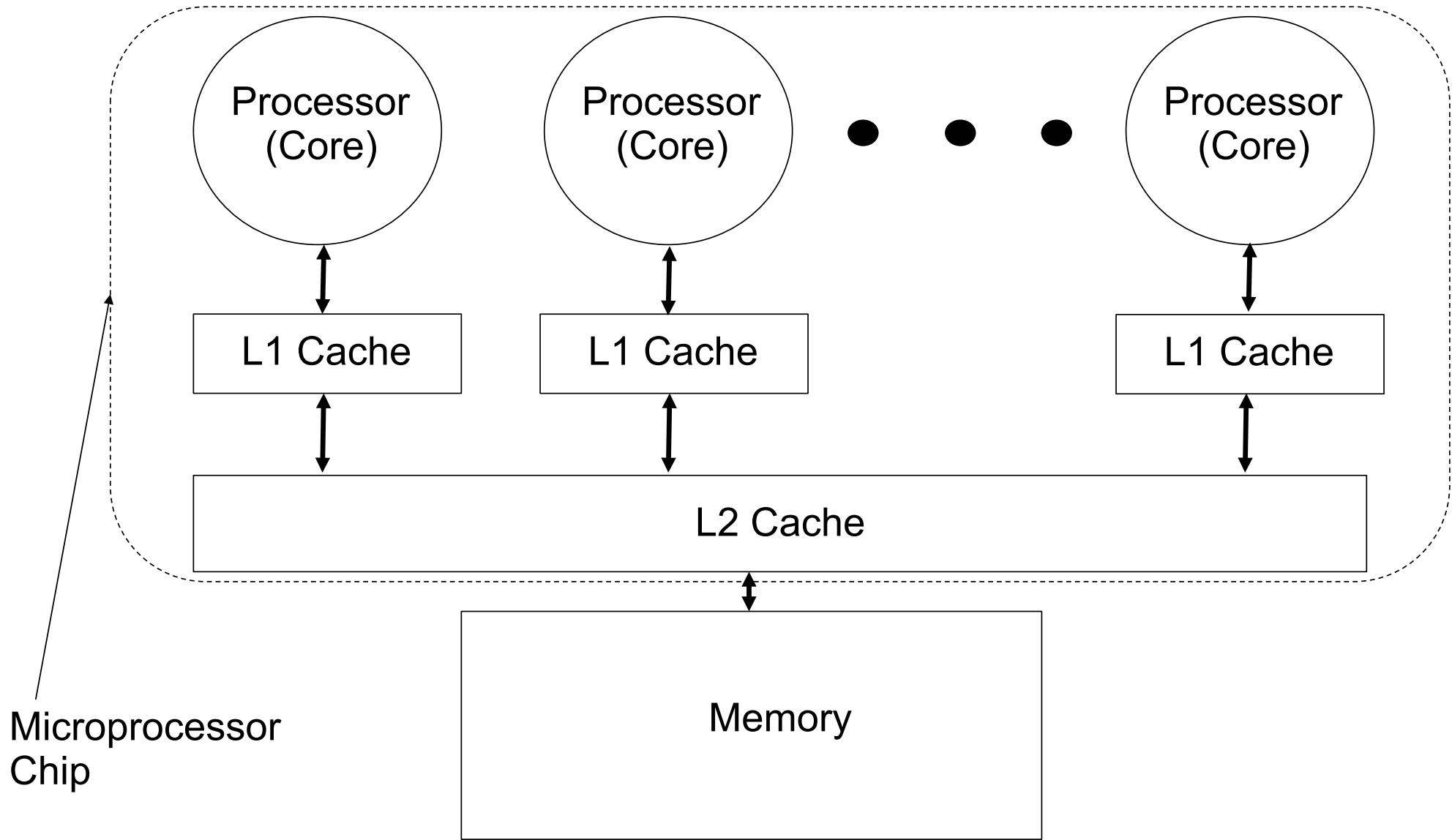
Cycle: 12



Loop

```
L.S F0, 0(R1)
L.S F1, 0(R2)
➔ ADD.S F2, F1, F0
S.S F2, 0(R1)
ADDI R1, R1, #4
ADDI R2, R2, #4
SUBI R3, R3, #1
BNEZ R3, Loop
```

# Multicore Computers





# Blended Learning

Lectures on youtube

9 interactive sessions, flipped class-room style

5 problem solution sessions

1 design exploration project

***Textbook:***

***Parallel Computer Organization and Design***

***Dubois, Annavaram, Stenström***

