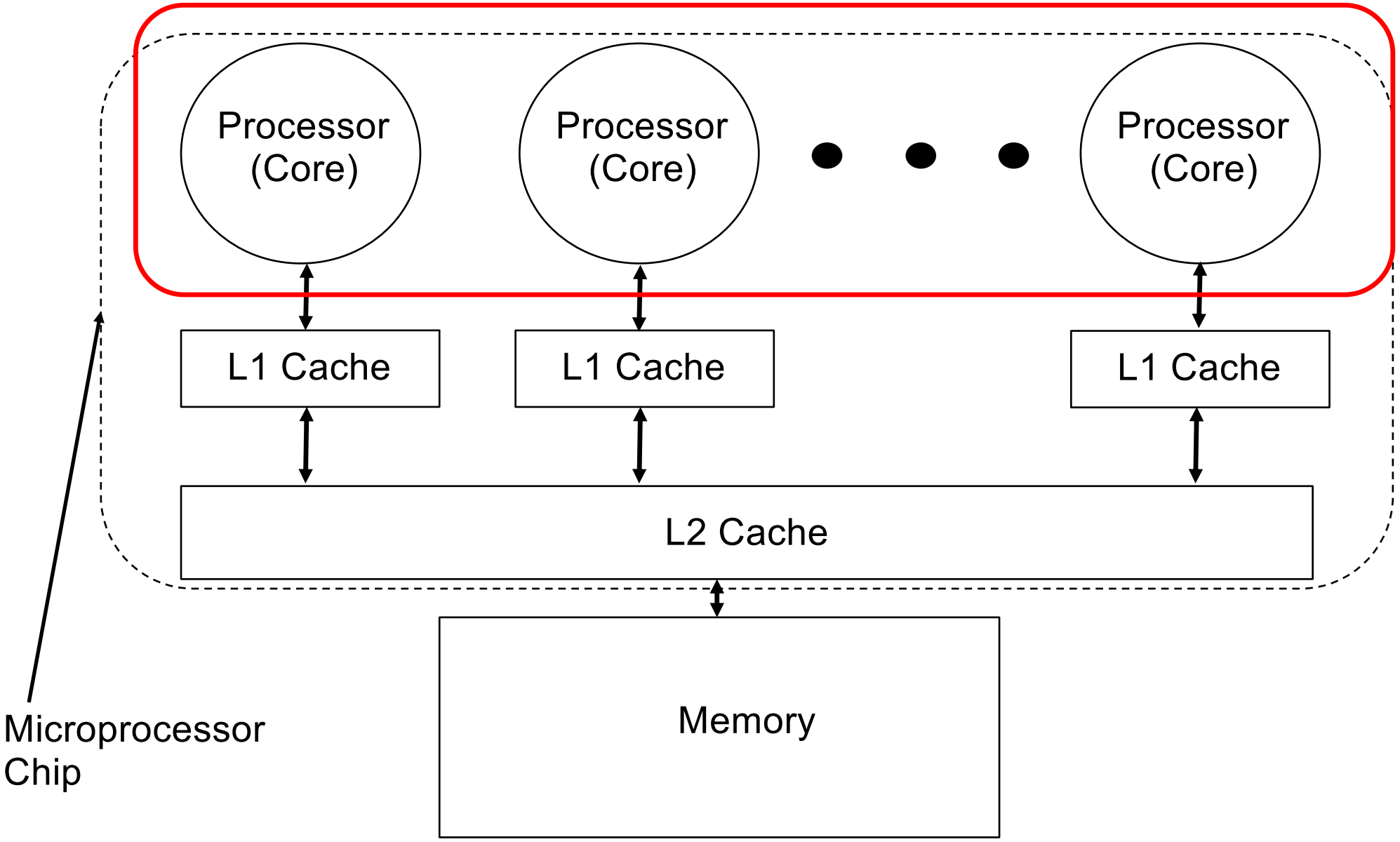
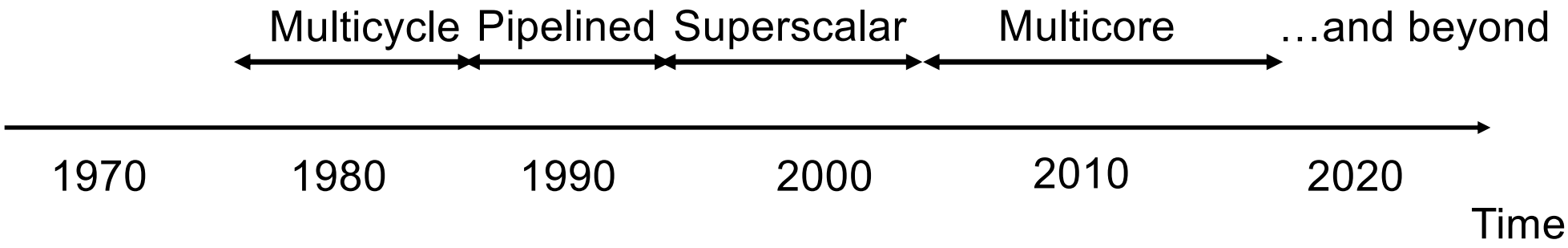
The Evolution of Microprocessors

Per Stenström

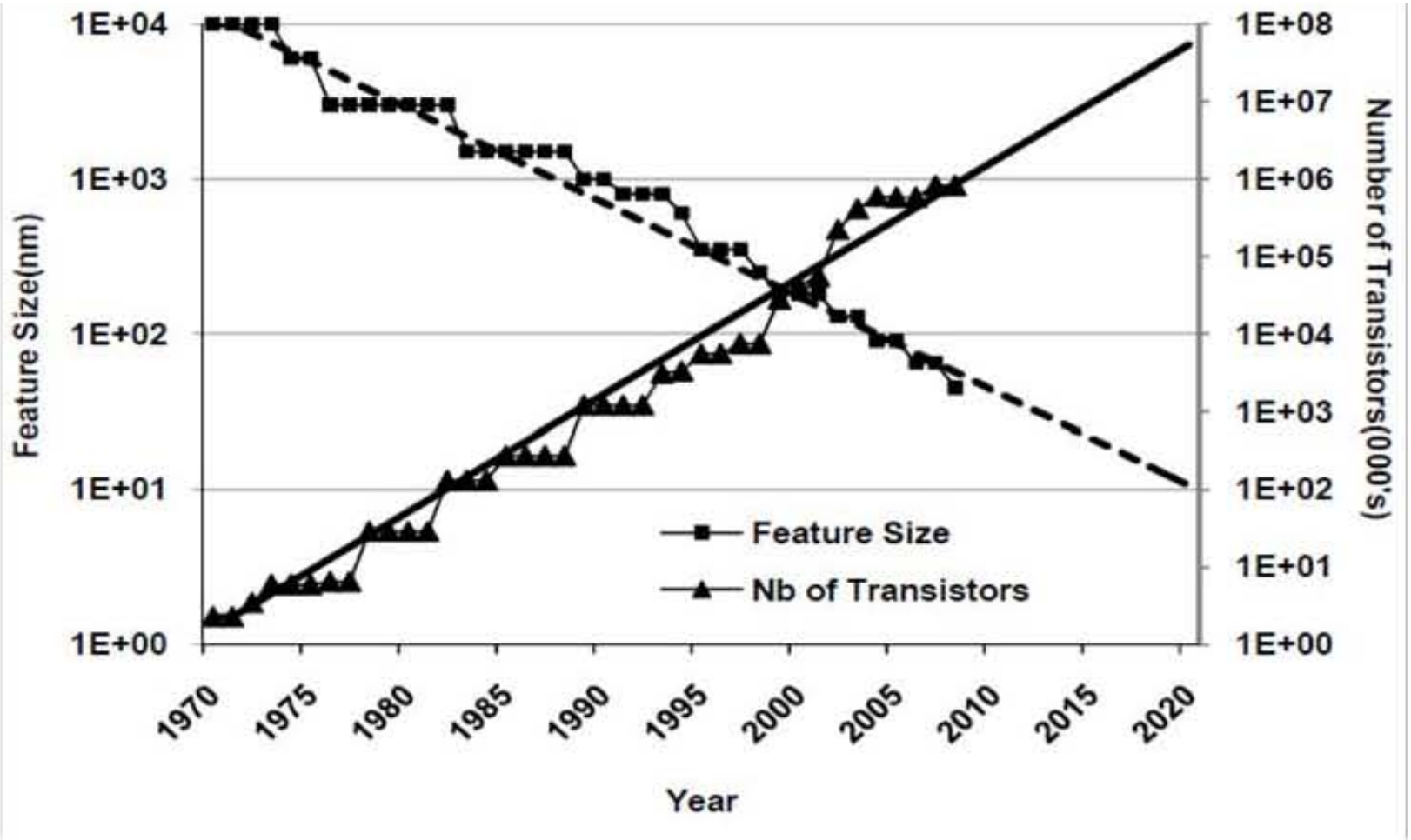

Processor
(Core)
Processor
(Core)
Processor
(Core)
L1 Cache
L1 Cache
L1 Cache
L2 Cache
Memory
Microprocessor
Chip

Evolution of Microprocessors

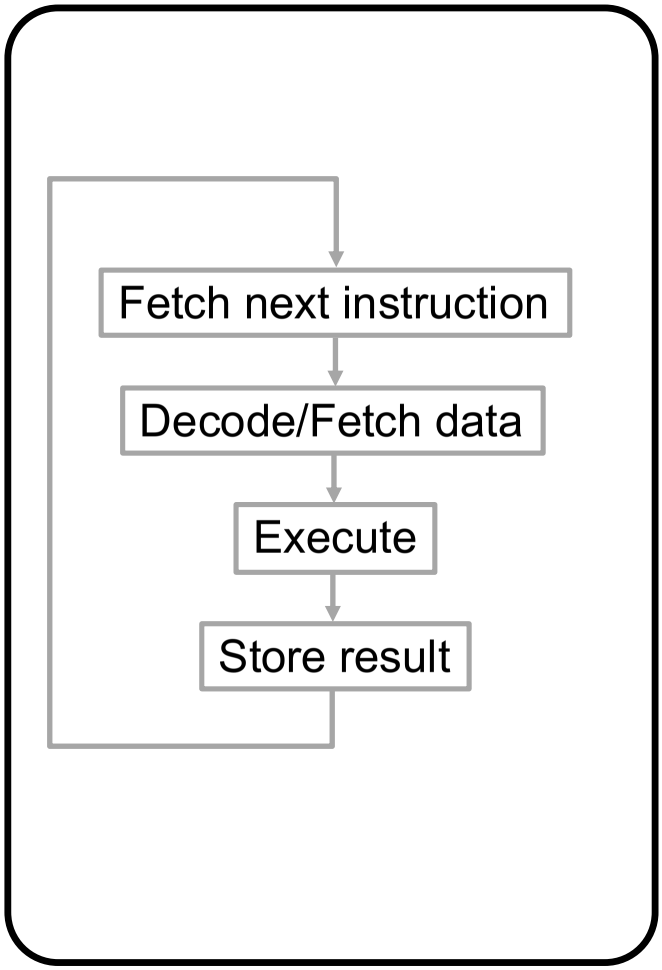


Key driver: Moore's Law

Moore's Law at Work



Multicycle Computers


Fetch next instruction
Decode/Fetch data
Execute
Store result

Instruction Category 1: ALU Instructions

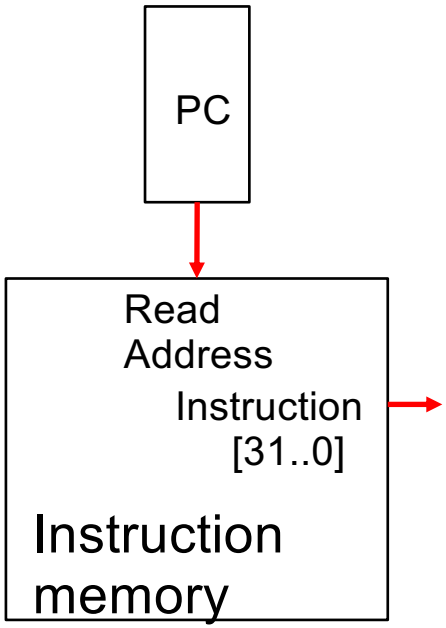
Opcode	Assembly code	Meaning	Comments
ADD,SUB,	ADD R1,R2,R3	R1=(R2) + (R3)	Integer operations
ADDI, SUBI	ADDI R1,R2,#3	R1=(R2)+3	Immediate
AND, OR, XOR	AND R1,R2,R3	R1=(R2).AND.(R3)	bit-wise logical AND,OR, XOR
SLT	SLT R1,R2,R3	If (R2)<(R3) then R1=1 else R1=0	Test R2, R3 outcome in R1
ADD.D, SUB.D,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

Instruction Category 2: Memory Instructions

Opcode	Assembly code	Meaning	Comments
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

Instruction Category 3:Control Instructions

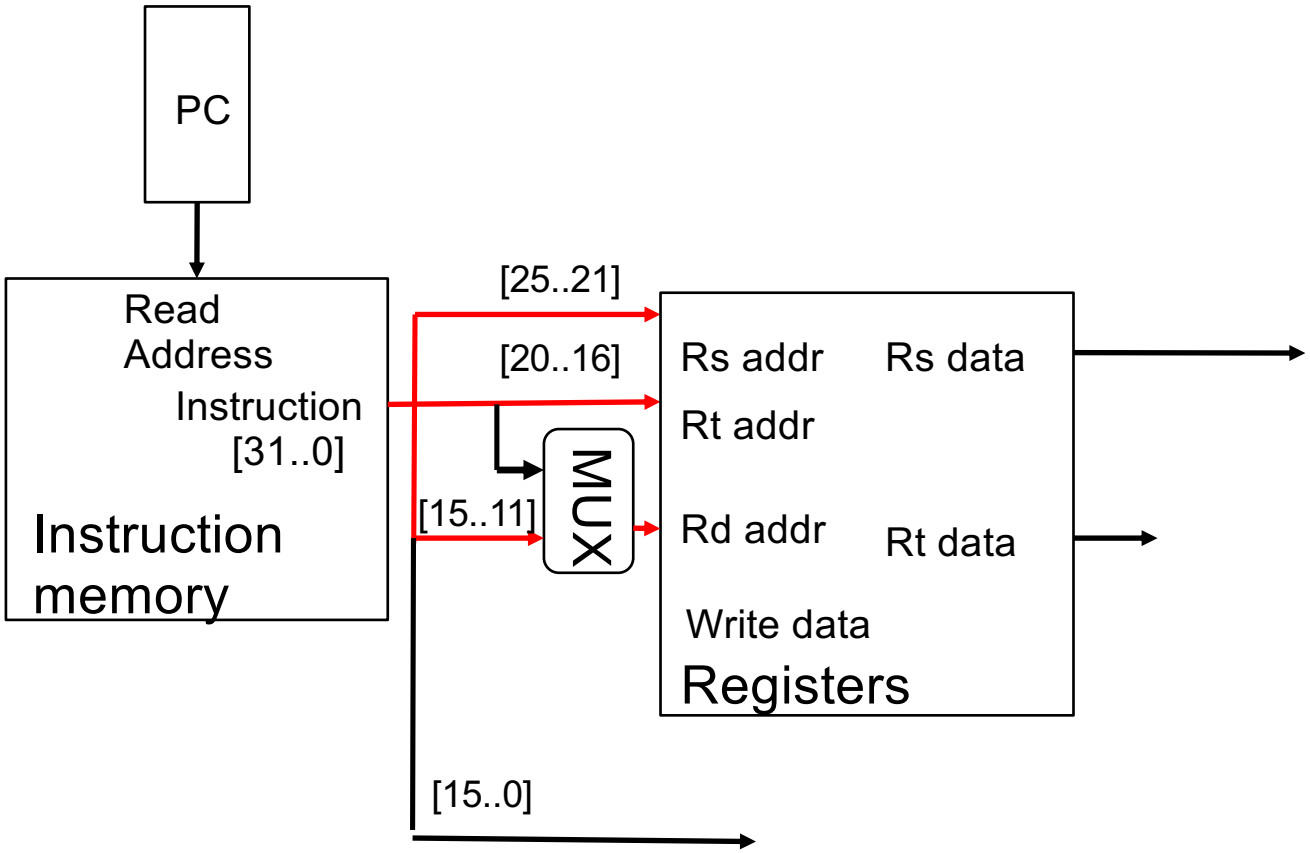
Opcode	Assembly code	Meaning	Comments
BEQZ, BNEZ	BEQZ R1,Label	If (R1)=0 then PC=Label	Conditional branch-equal 0/not equal 0
BEQ, BNE	BEQ R1,R2,Label	If (R1)=(R2) PC=Label	Conditional branch-equal/not equal
J	J Target	PC=Target	Target is an immediate field
JR	JR R1	PC=(R1)	Target is in register
JAL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


PC
Read
Address
Instruction
[31..0]
Instruction
memory

ALU

Memory transfer

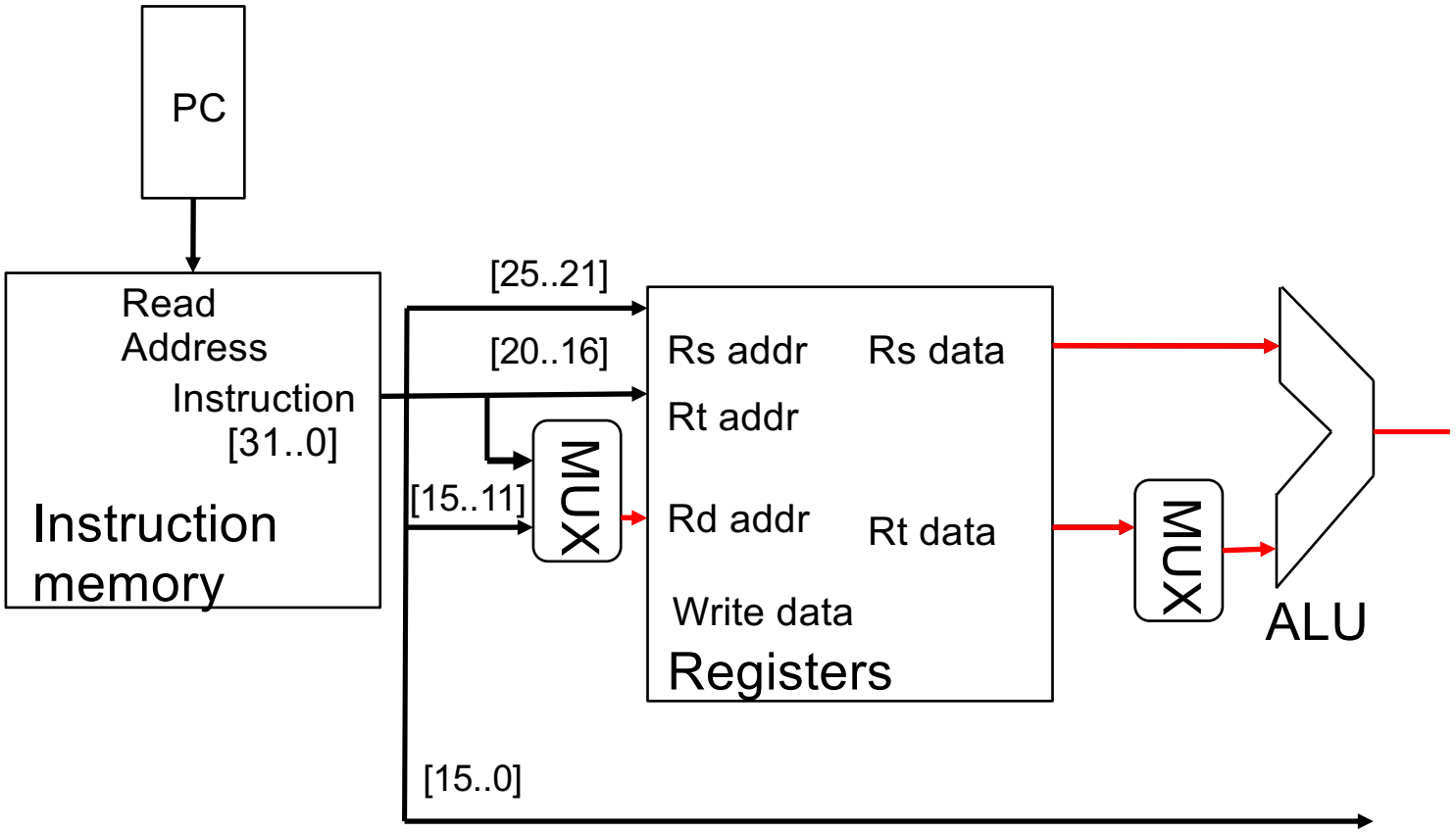
Control

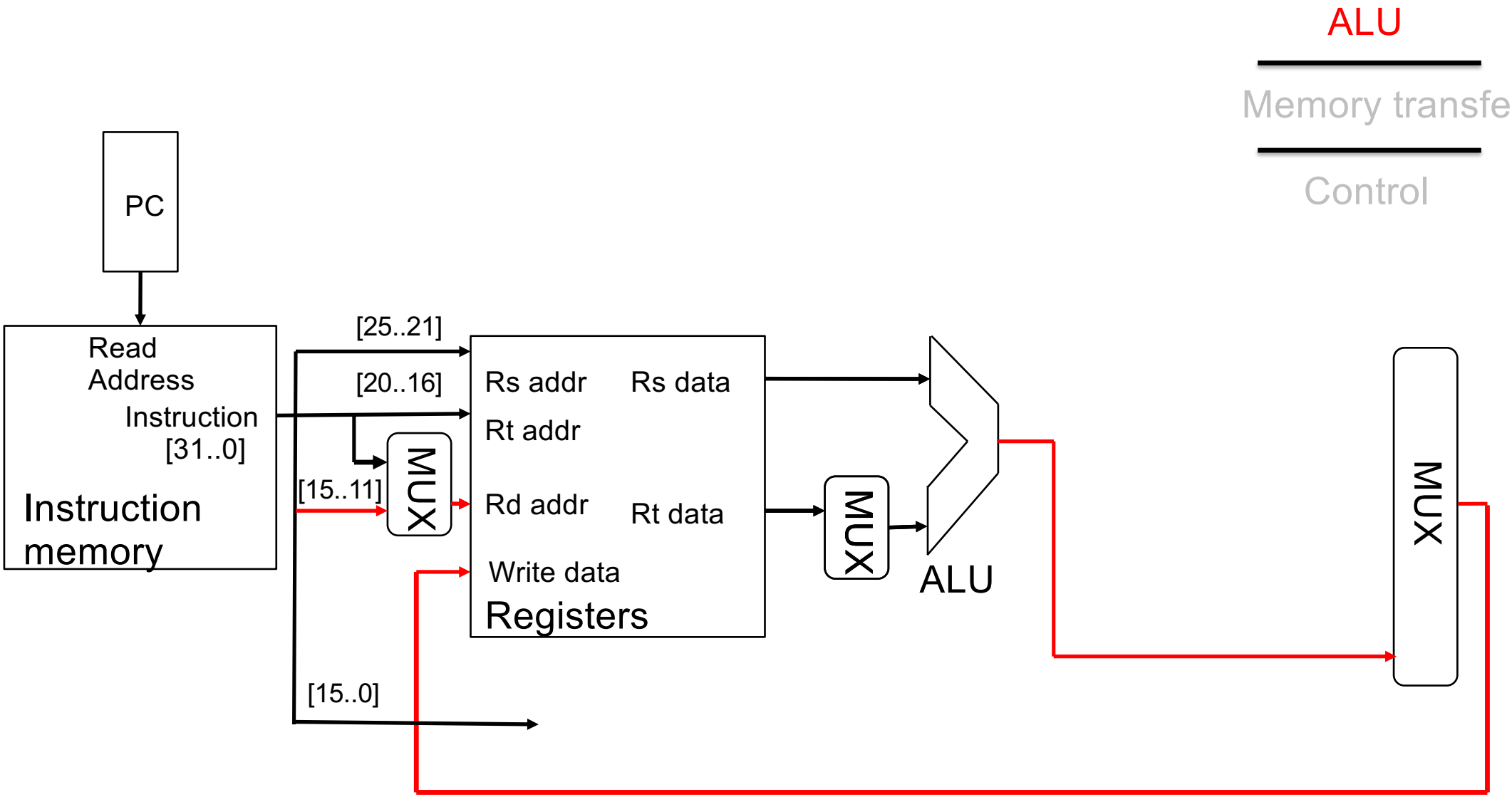


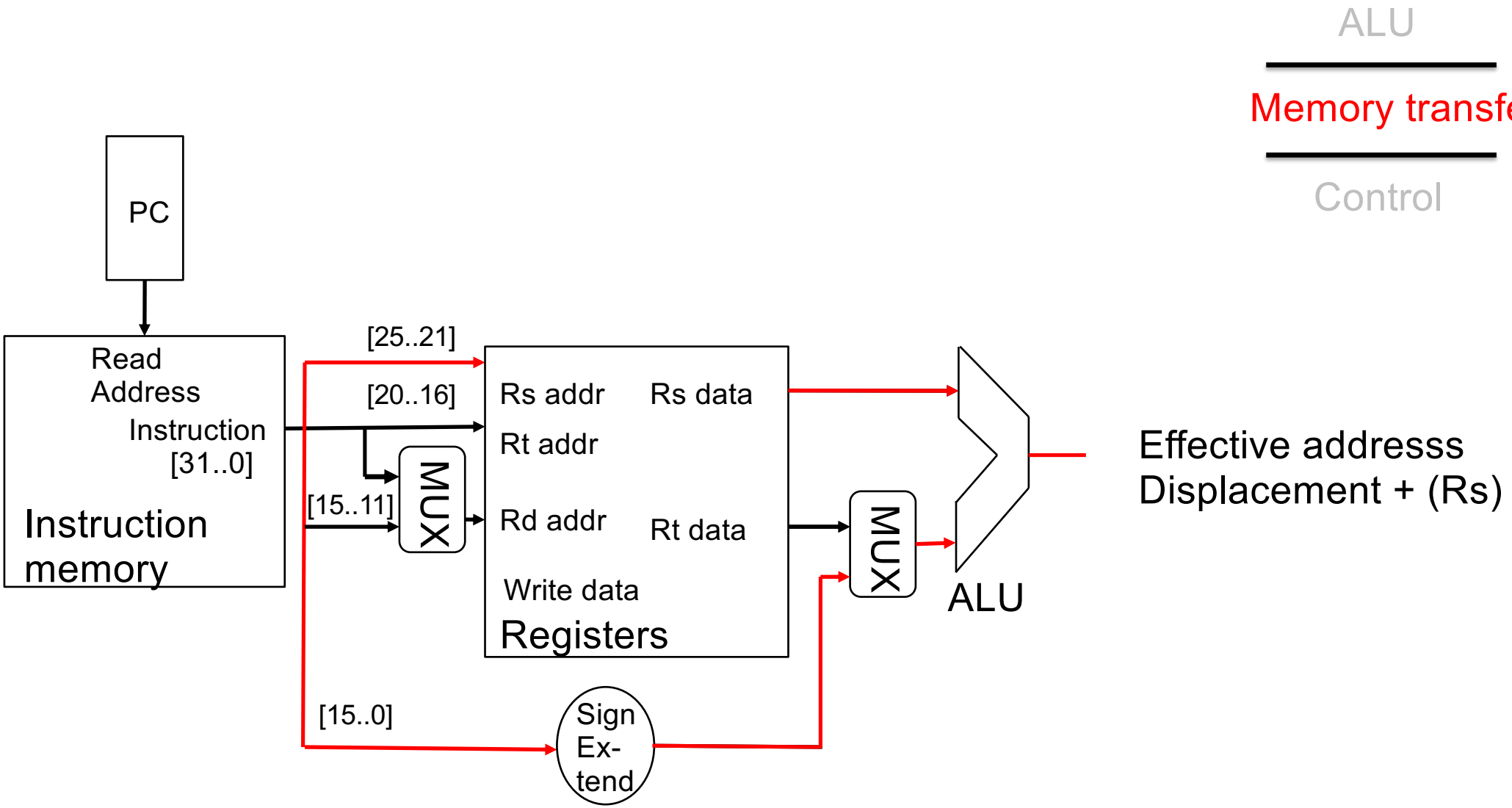
ALU

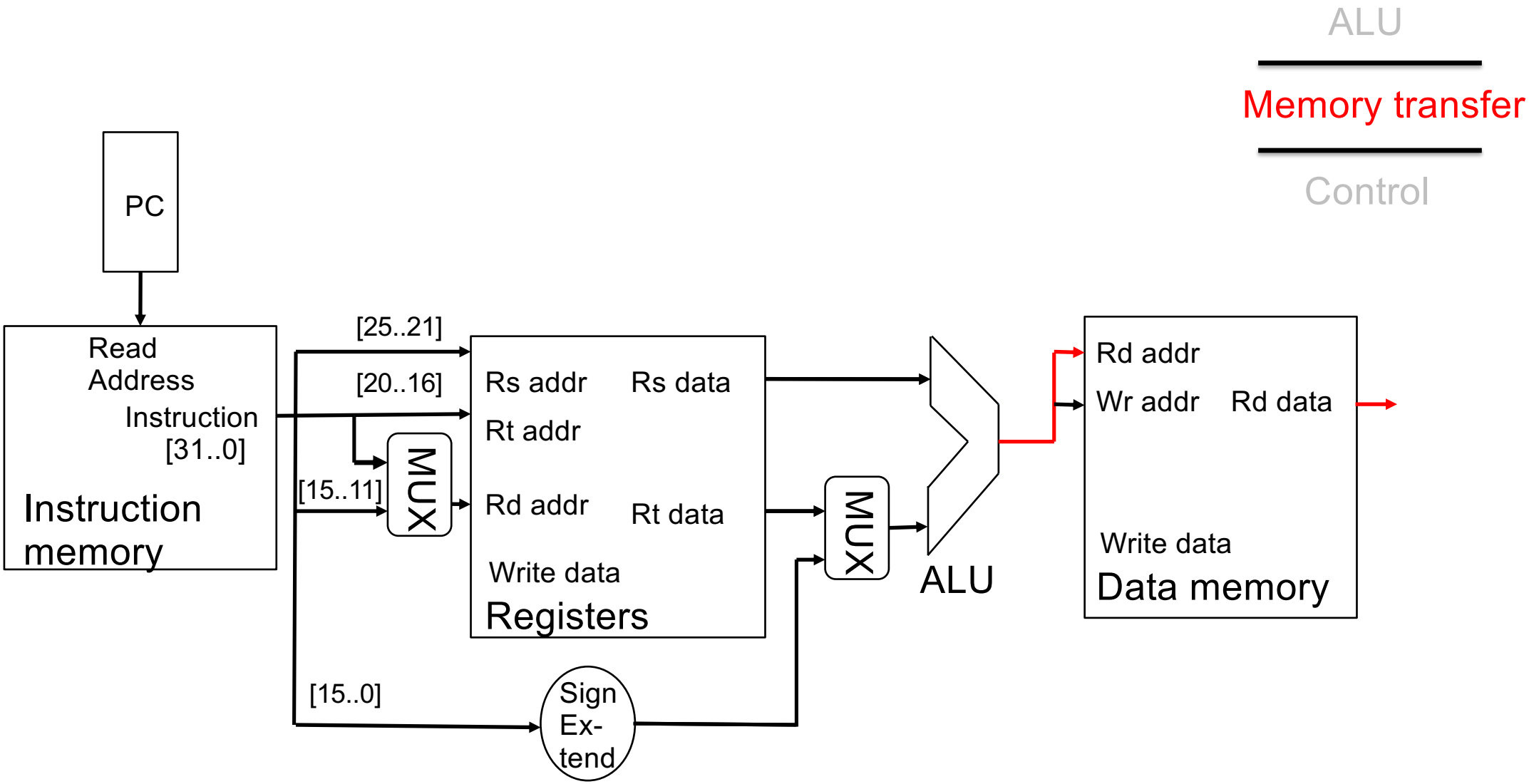
Memory transfer

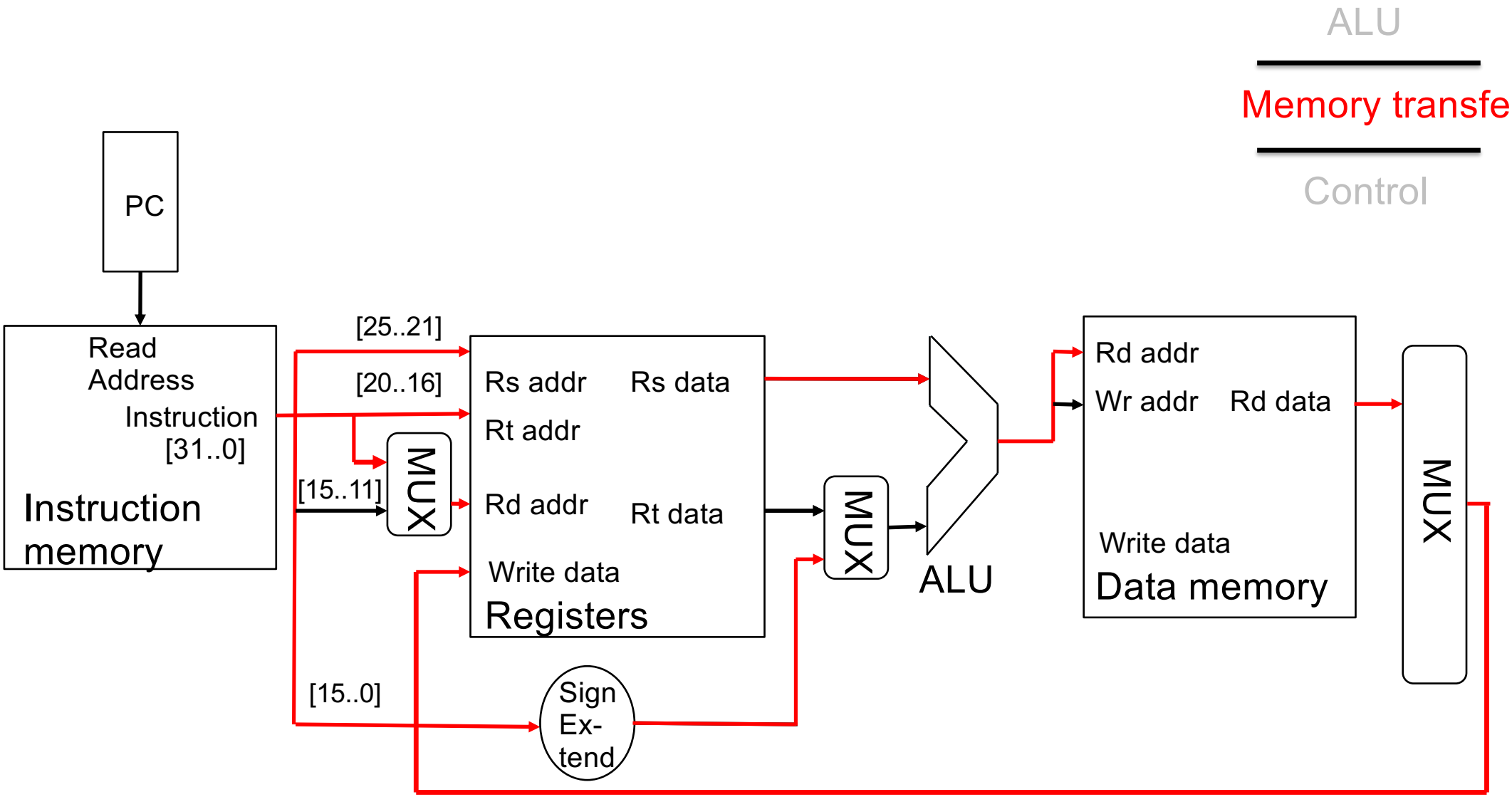
Control

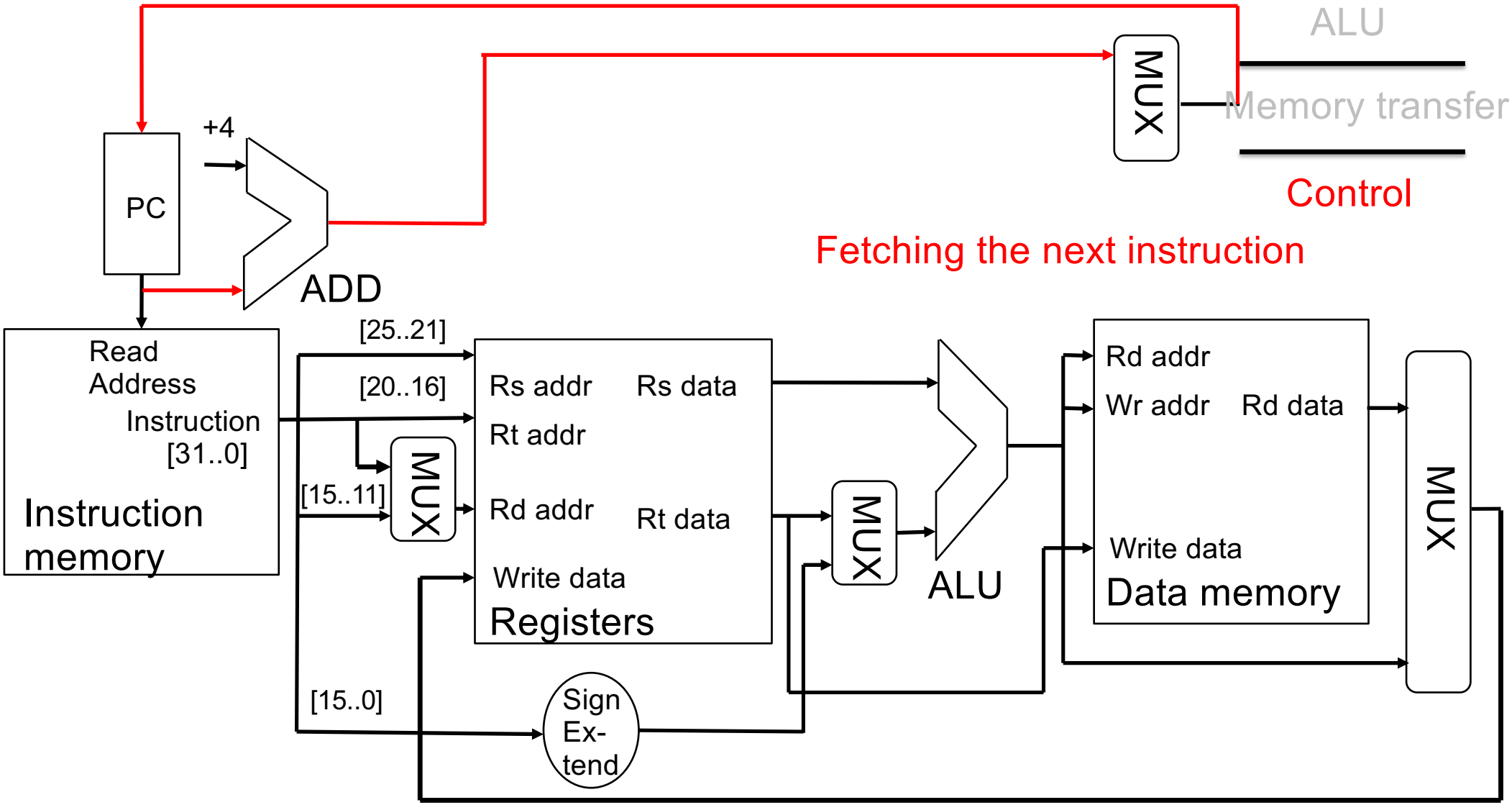


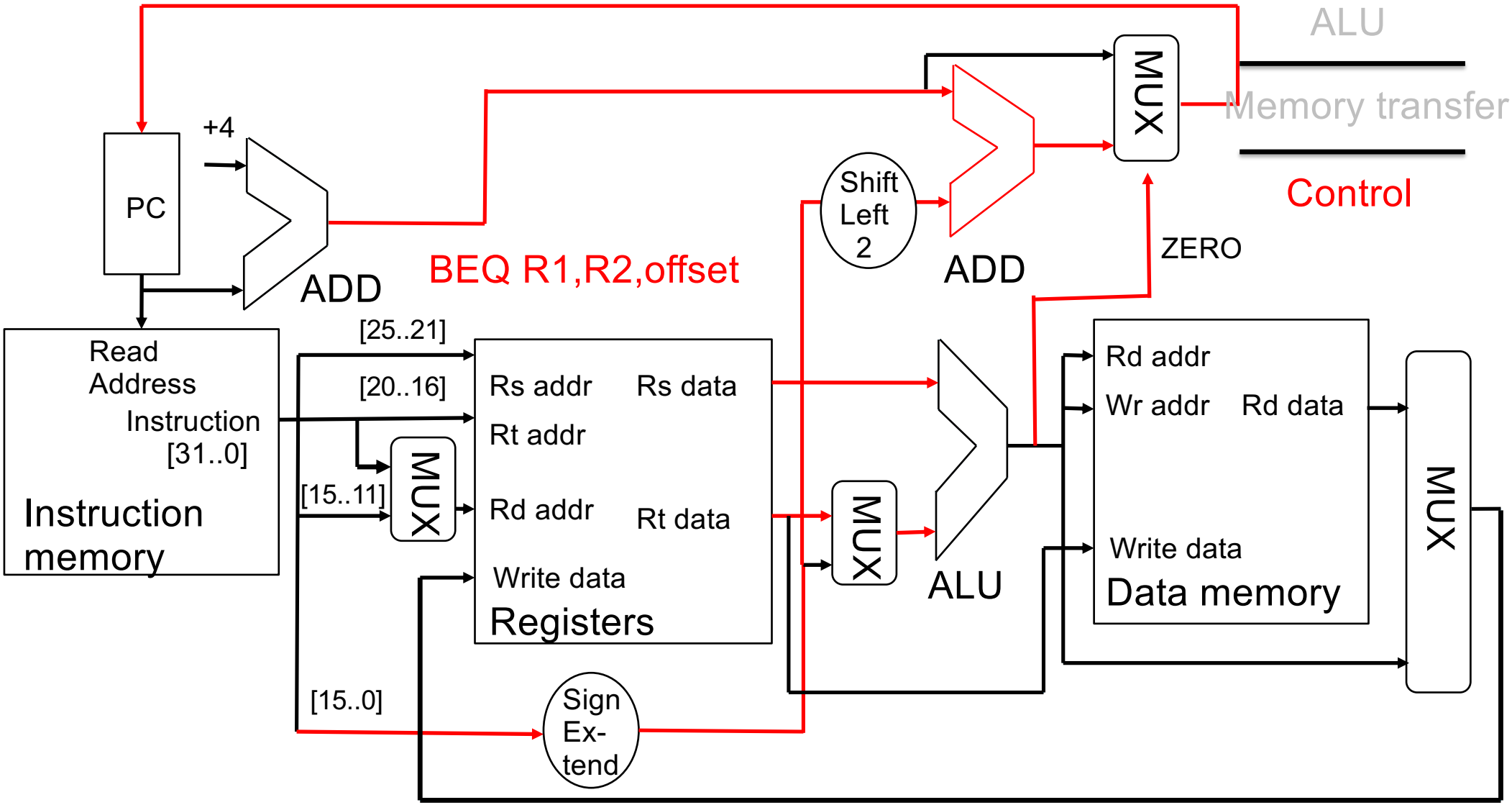

ALU
Memory transfer
Control
PC
Read
Address
Instruction
[31..0]
Instruction
memory
[25..21]
[20..16]
[15..11]
[15..0]
MUX
Rs addr
Rs data
Rt addr
Rd addr
Rt data
Write data
Registers
MUX
ALU
MUX

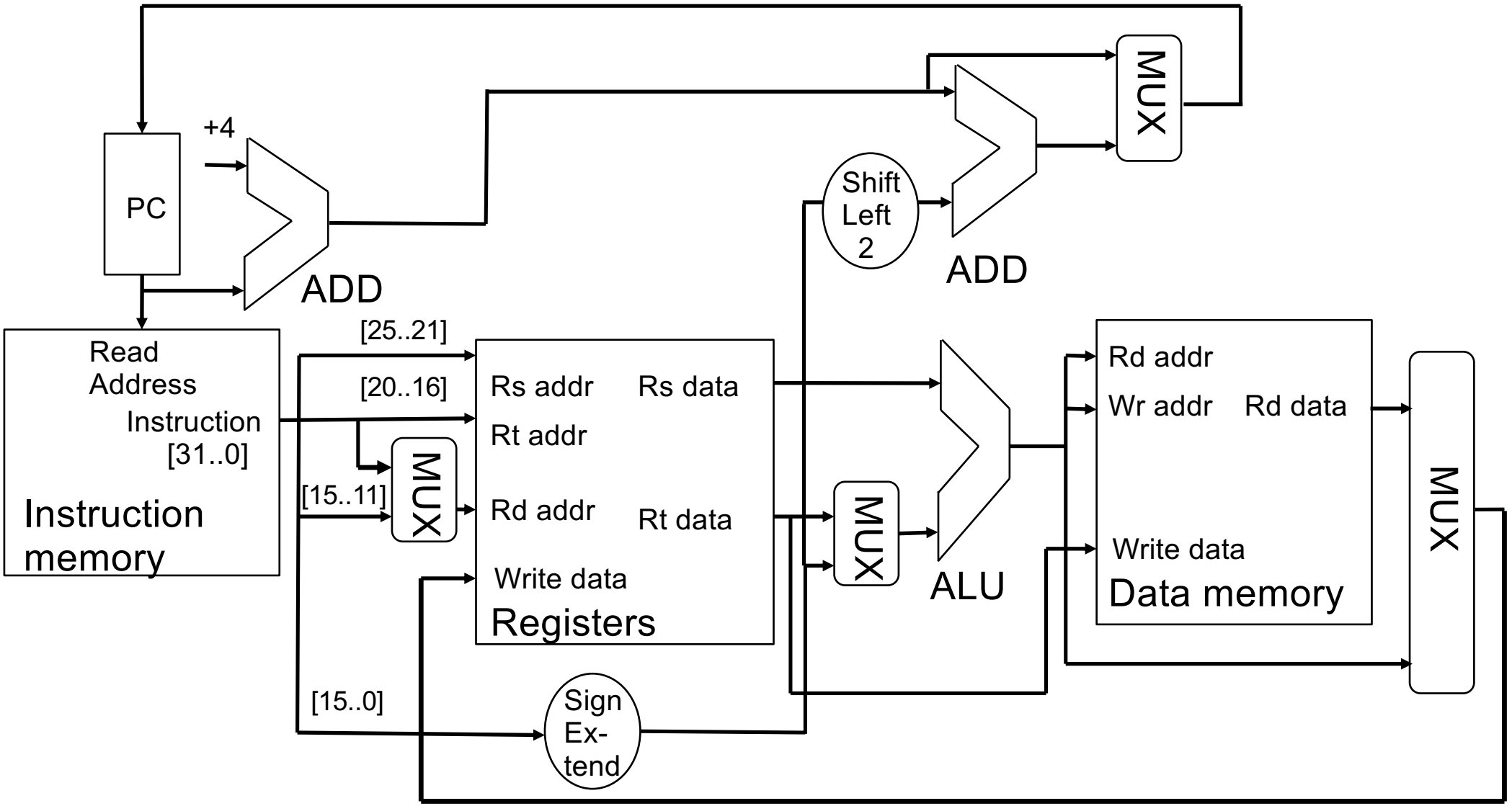

ALU
Memory transfer
Control
PC
Read
Address
Instruction
[31..0]
Instruction
memory
[25..21]
[20..16]
[15..11]
MUX
Rs addr
Rs data
Rt addr
Rd addr
Rt data
Write data
Registers
MUX
ALU
Effective addresss
Displacement + (Rs)
[15..0]
Sign
Ex-
tend

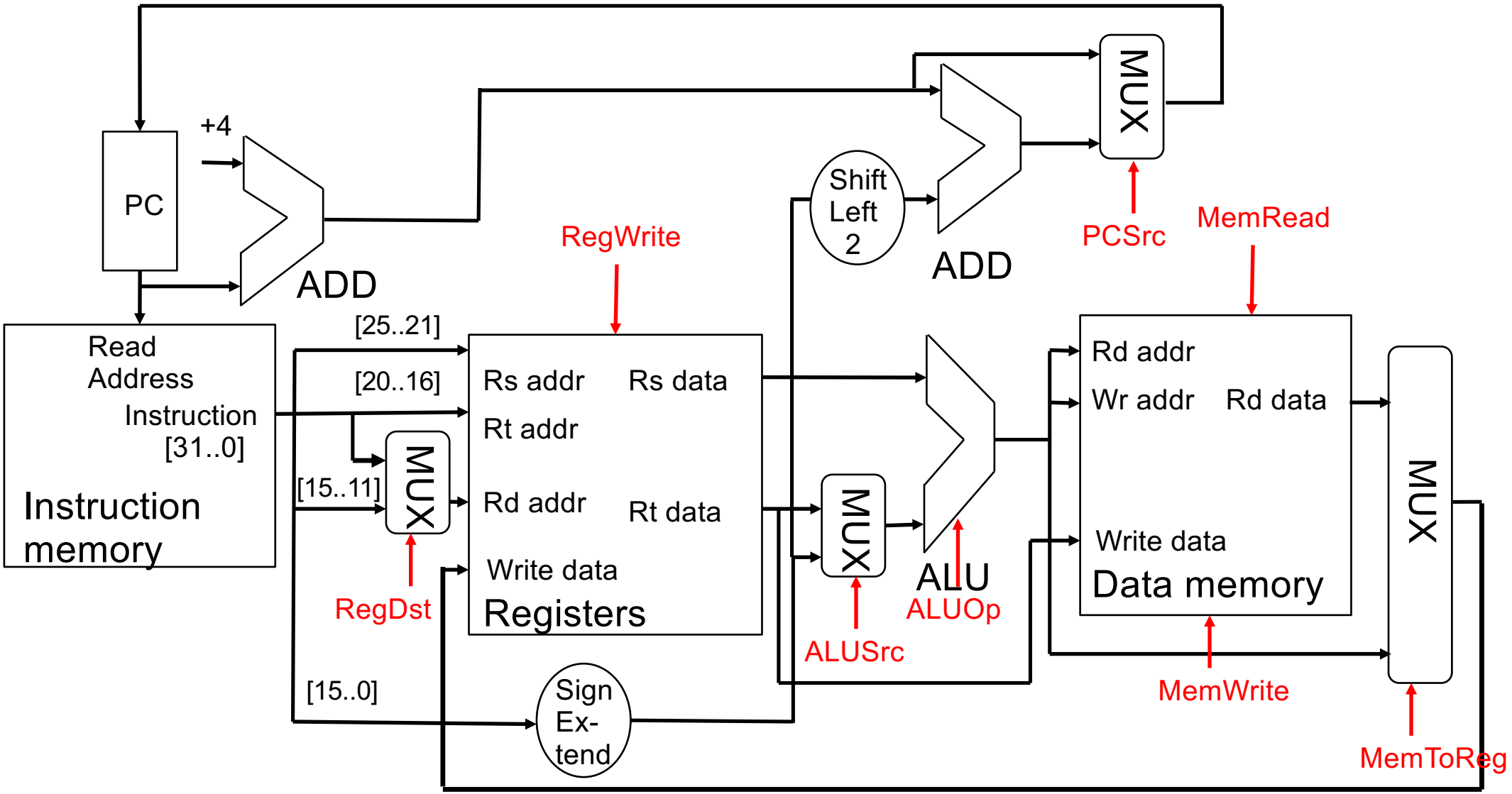

ALU
Memory transfer
Control
PC
Read
Address
Instruction
[31..0]
Instruction
memory
[25..21]
[20..16]
[15..11]
MUX
Rs addr
Rs data
Rt addr
Rd addr
Rt data
Write data
Registers
[15..0]
Sign
Ex-
tend
MUX
ALU
Rd addr
Wr addr
Rd data
Write data
Data memory

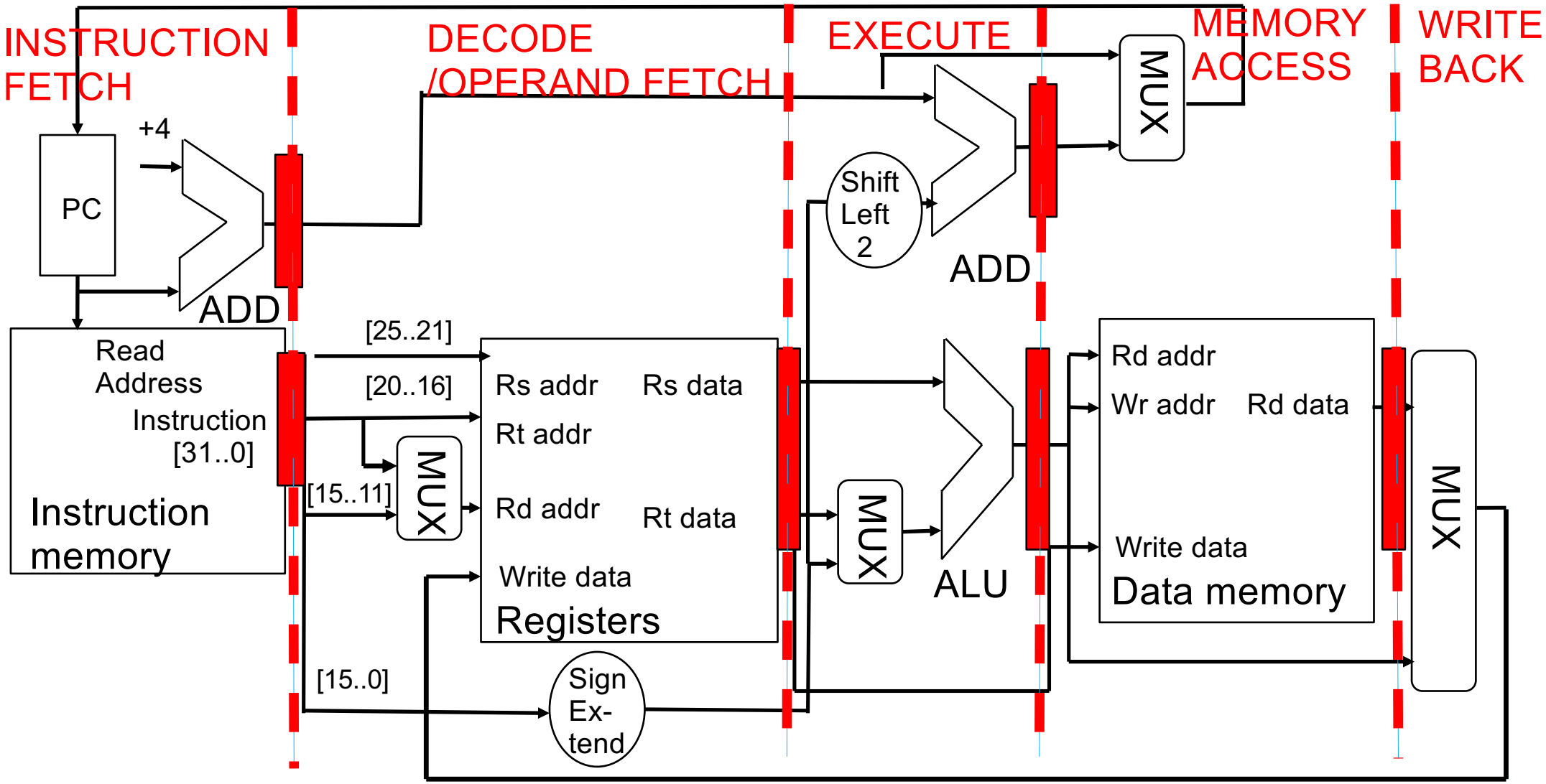

ALU
Memory transfer
Control
PC
Read
Address
Instruction
[31..0]
Instruction
memory
[25..21]
[20..16]
[15..11]
MUX
Rs addr
Rs data
Rt addr
Rd addr
Rt data
Write data
Registers
[15..0]
Sign
Ex-
tend
MUX
ALU
Rd addr
Wr addr
Rd data
Write data
Data memory
MUX


ALU
MUX
Memory transfer
Control
Fetching the next instruction
+4
PC
ADD
Read
Address
Instruction
[31..0]
Instruction
memory
[25..21]
[20..16]
[15..11]
MUX
Rs addr
Rs data
Rt addr
Rd addr
Rt data
Write data
Registers
MUX
ALU
Rd addr
Wr addr
Rd data
Write data
Data memory
MUX
[15..0]
Sign
Ex-
tend


ALU
Memory transfer
Control
MUX
+4
PC
ADD
BEQ R1,R2,offset
Shift
Left
2
ADD
ZERO
Read
Address
Instruction
[31..0]
Instruction
memory
[25..21]
[20..16]
[15..11]
MUX
Rs addr
Rs data
Rt addr
Rd addr
Rt data
Write data
Registers
MUX
ALU
Rd addr
Wr addr
Rd data
Write data
Data memory
MUX
[15..0]
Sign
Ex-
tend

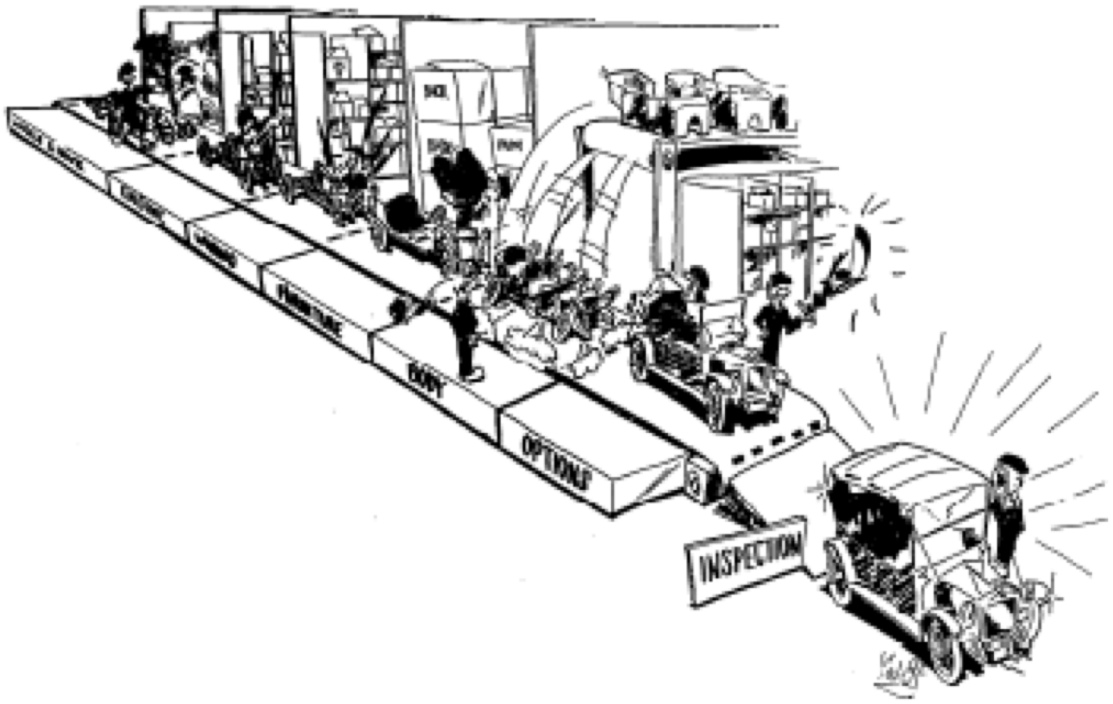

PC
+4
ADD
MUX
Shift
Left
2
ADD
Read
Address
Instruction
[31..0]
Instruction
memory
[25..21]
[20..16]
[15..11]
MUX
Rs addr
Rs data
Rt addr
Rd addr
Rt data
Write data
Registers
MUX
ALU
Rd addr
Wr addr
Rd data
Write data
Data memory
MUX
[15..0]
Sign
Ex-
tend

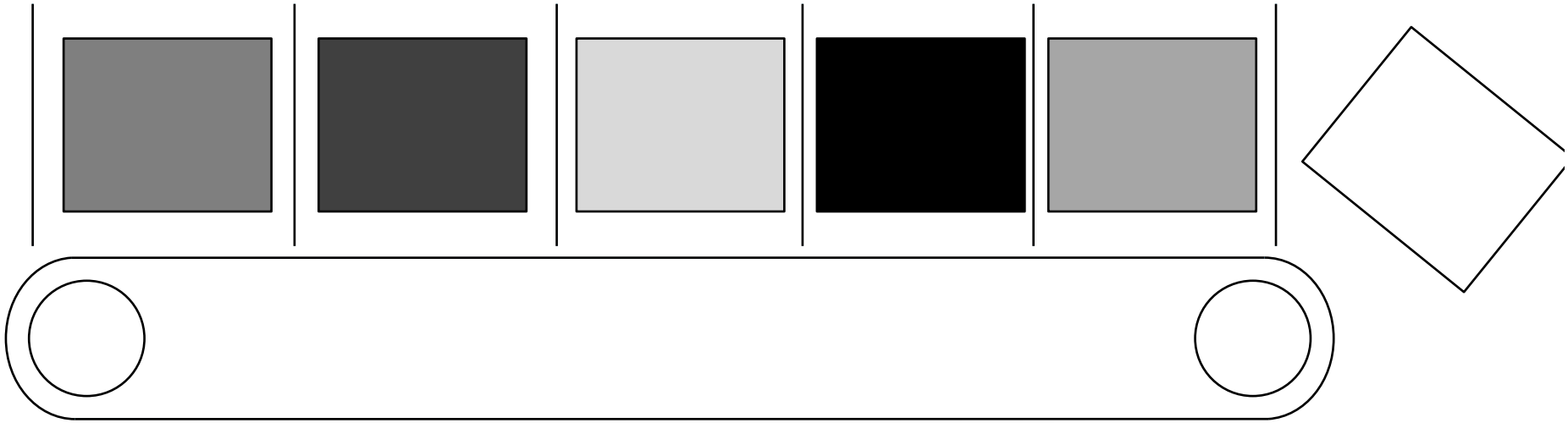

PC
+4
ADD
Read
Address
Instruction
[31..0]
Instruction
memory
[25..21]
[20..16]
[15..11]
MUX
RegDst
RegWrite
Rs addr
Rs data
Rt addr
Rd addr
Rt data
Write data
Registers
[15..0]
Sign
Ex-
tend
Shift
Left
2
ADD
MUX
PCSrc
MUX
ALUSrc
ALU
ALUOp
MemRead
Rd addr
Wr addr
Rd data
Write data
Data memory
MemWrite
MUX
MemToReg

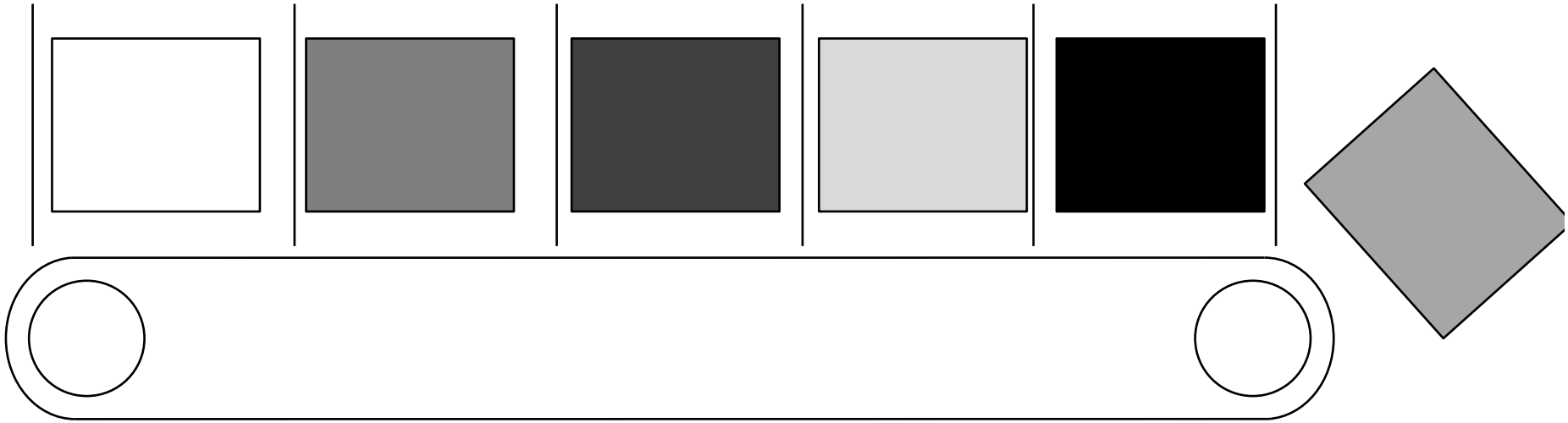

INSTRUCTION FETCH
DECODE /OPERAND FETCH
EXECUTE
MEMORY ACCESS
WRITE BACK
PC
+4
ADD
Read Address
Instruction [31..0]
Instruction memory
[25..21]
[20..16]
[15..11]
[15..0]
MUX
Rs addr
Rs data
Rt addr
Rd addr
Rt data
Write data
Registers
Sign Ex-tend
Shift Left 2
ADD
MUX
MUX
ALU
Rd addr
Wr addr
Rd data
Write data
Data memory
MUX

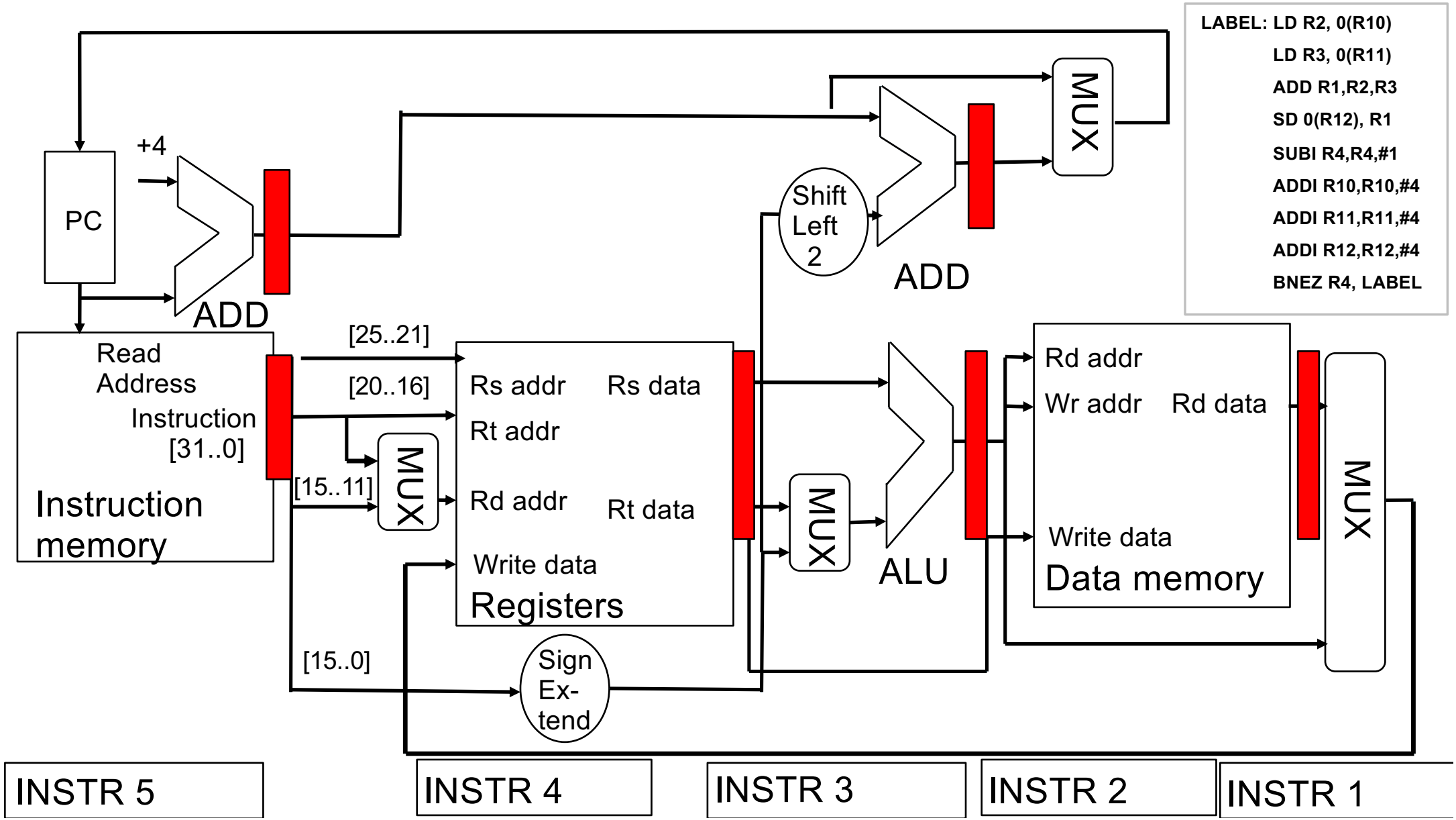
Pipelined ”Single-Cycle” Computer

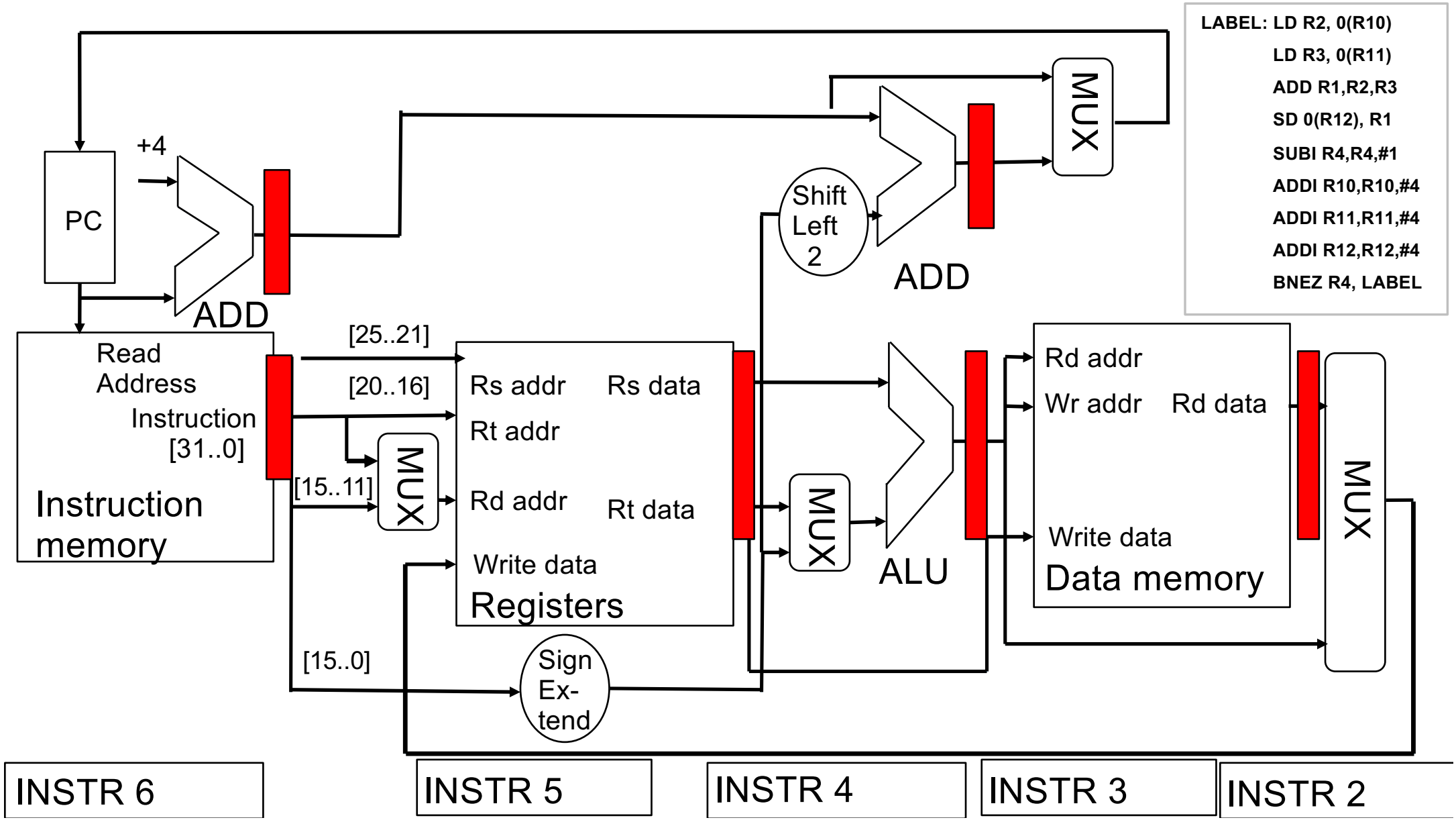
The Conveyor Belt

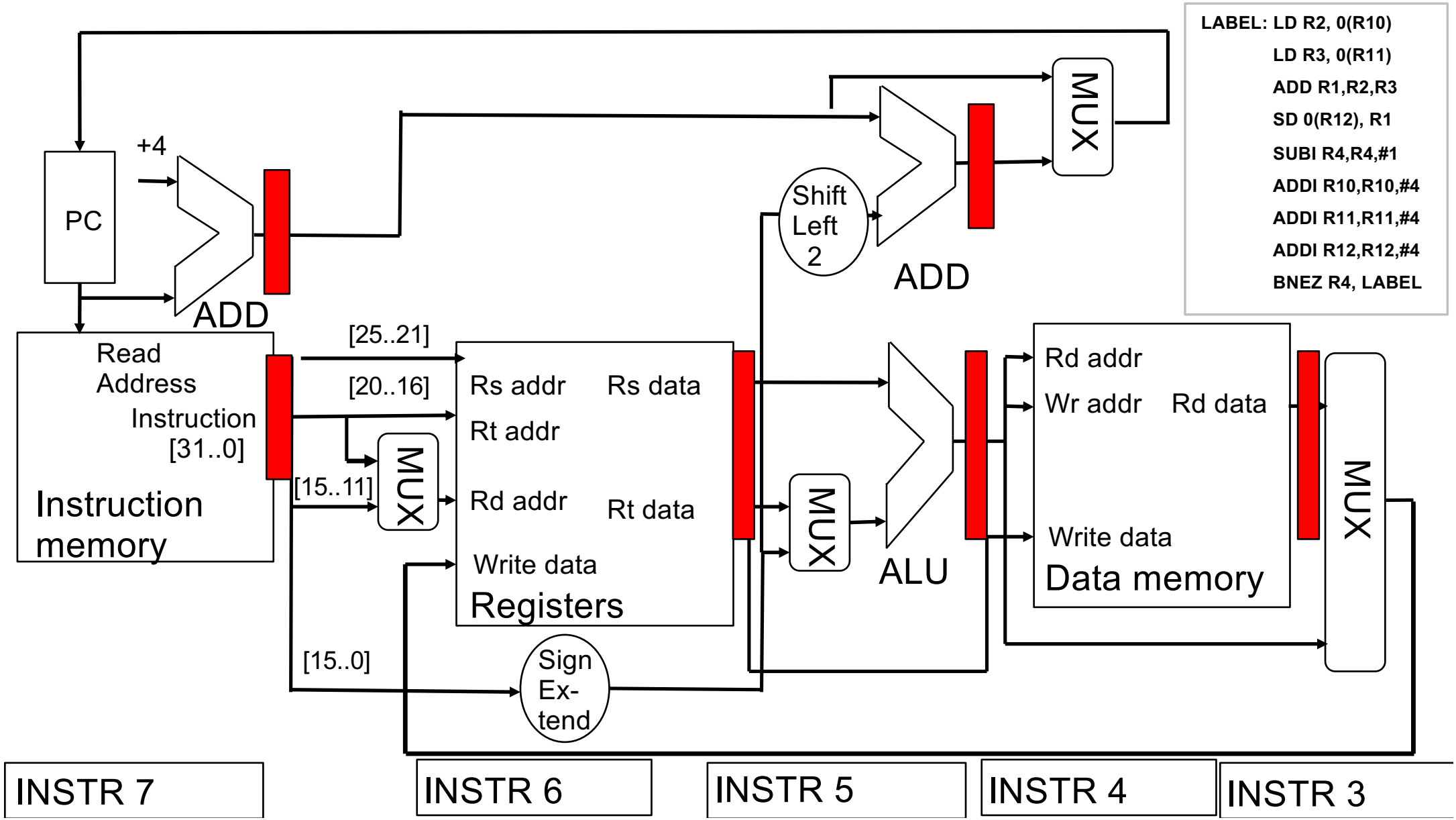


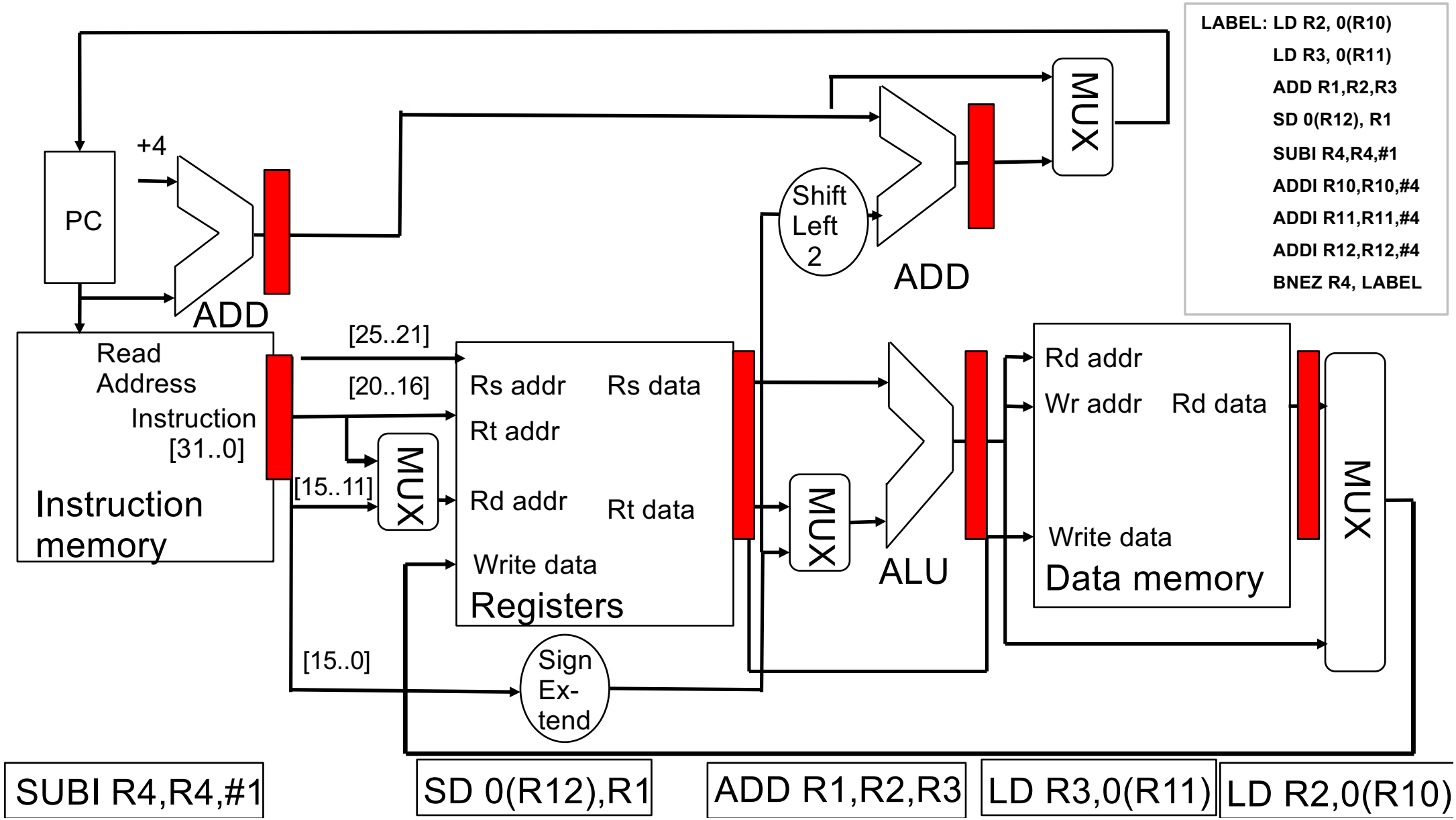


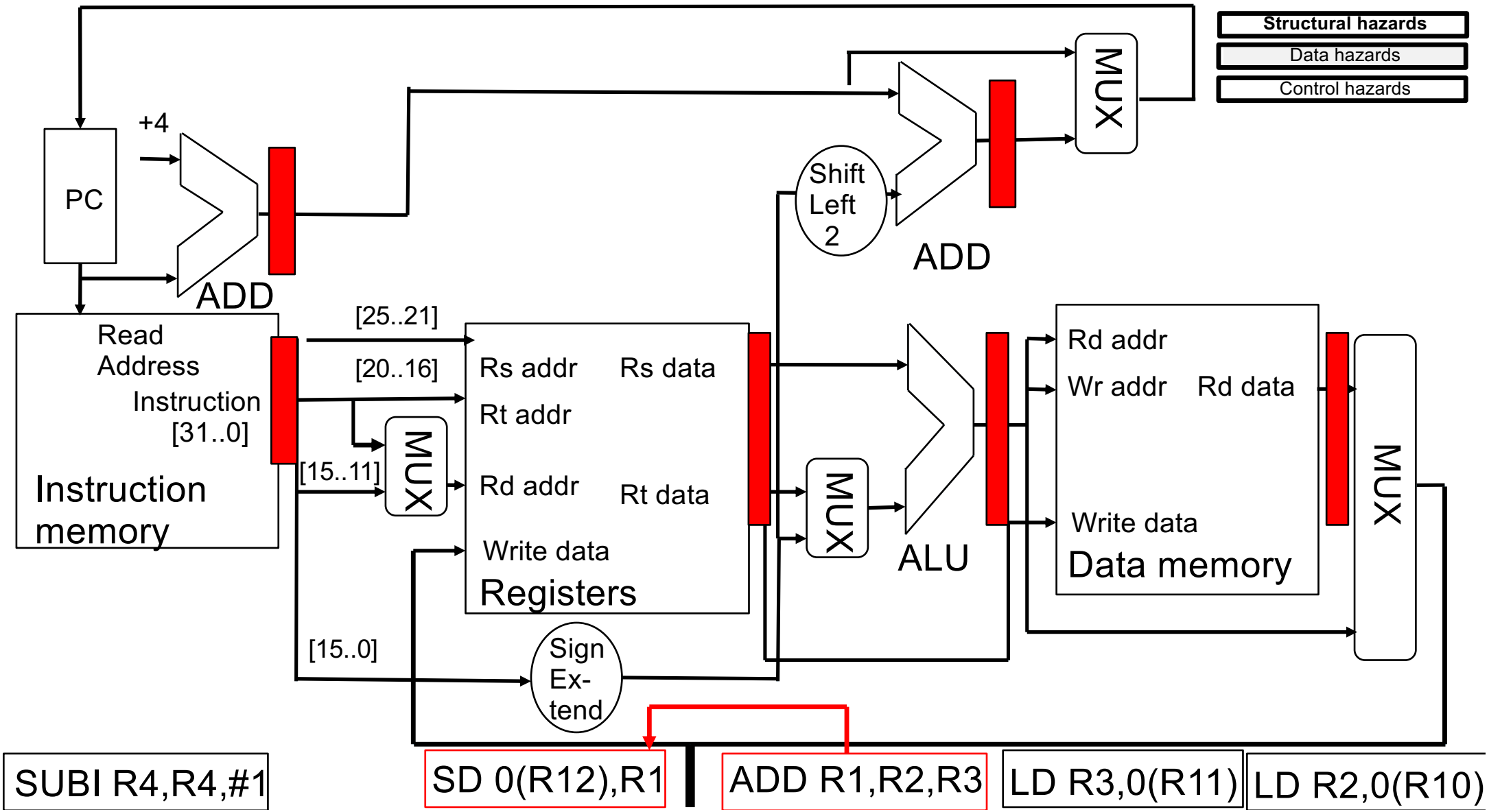


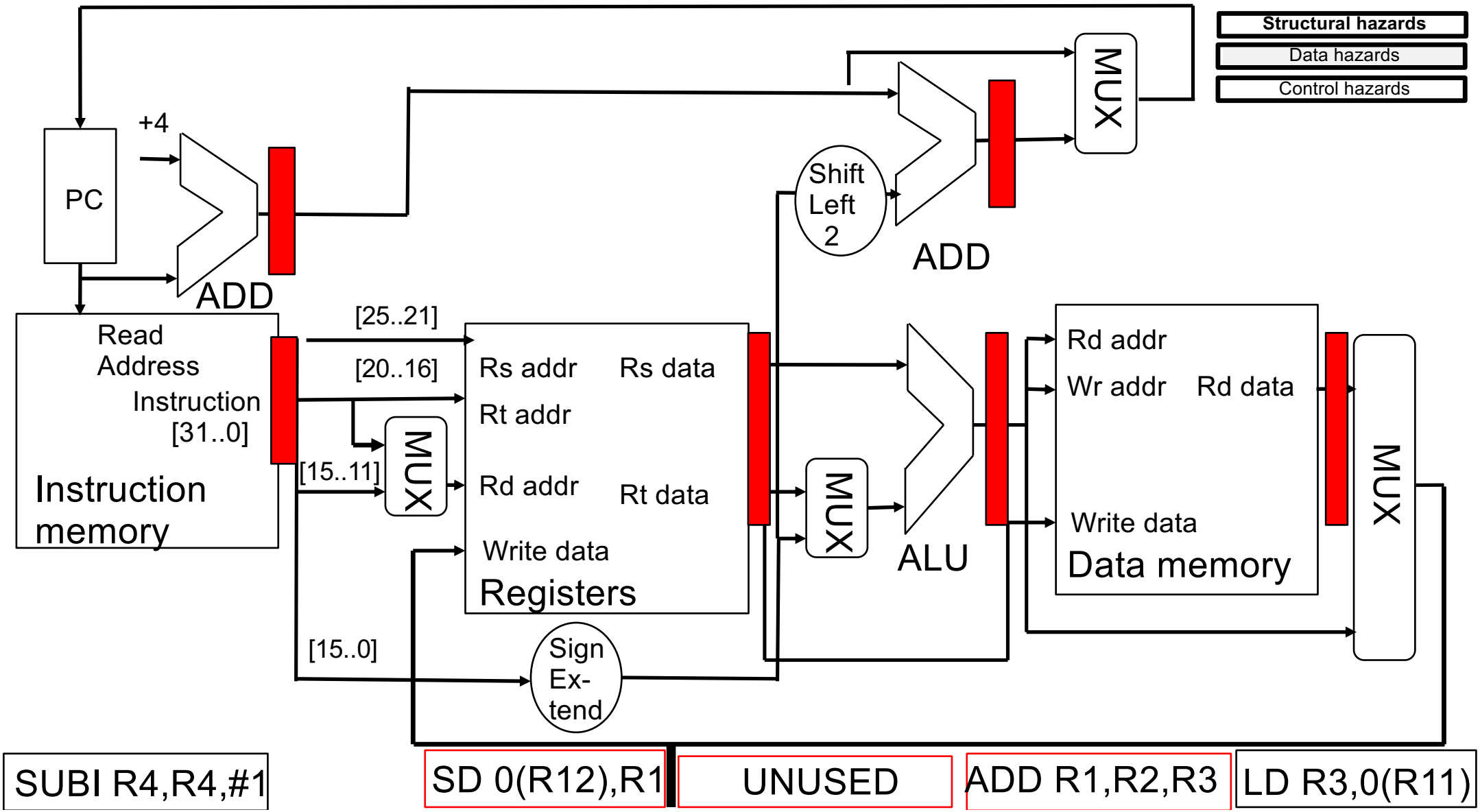

LABEL: LD R2, 0(R10)
LD R3, 0(R11)
ADD R1,R2,R3
SD 0(R12), R1
SUBI R4,R4,#1
ADDI R10,R10,#4
ADDI R11,R11,#4
ADDI R12,R12,#4
BNEZ R4, LABEL
PC
+4
ADD
MUX
Shift Left 2
ADD
Read Address
Instruction [31..0]
Instruction memory
[25..21]
[20..16]
[15..11]
MUX
Rs addr
Rs data
Rt addr
Rd addr
Rt data
Write data
Registers
[15..0]
Sign Ex-tend
MUX
ALU
Rd addr
Wr addr
Rd data
Write data
Data memory
MUX
INSTR 5
INSTR 4
INSTR 3
INSTR 2
INSTR 1

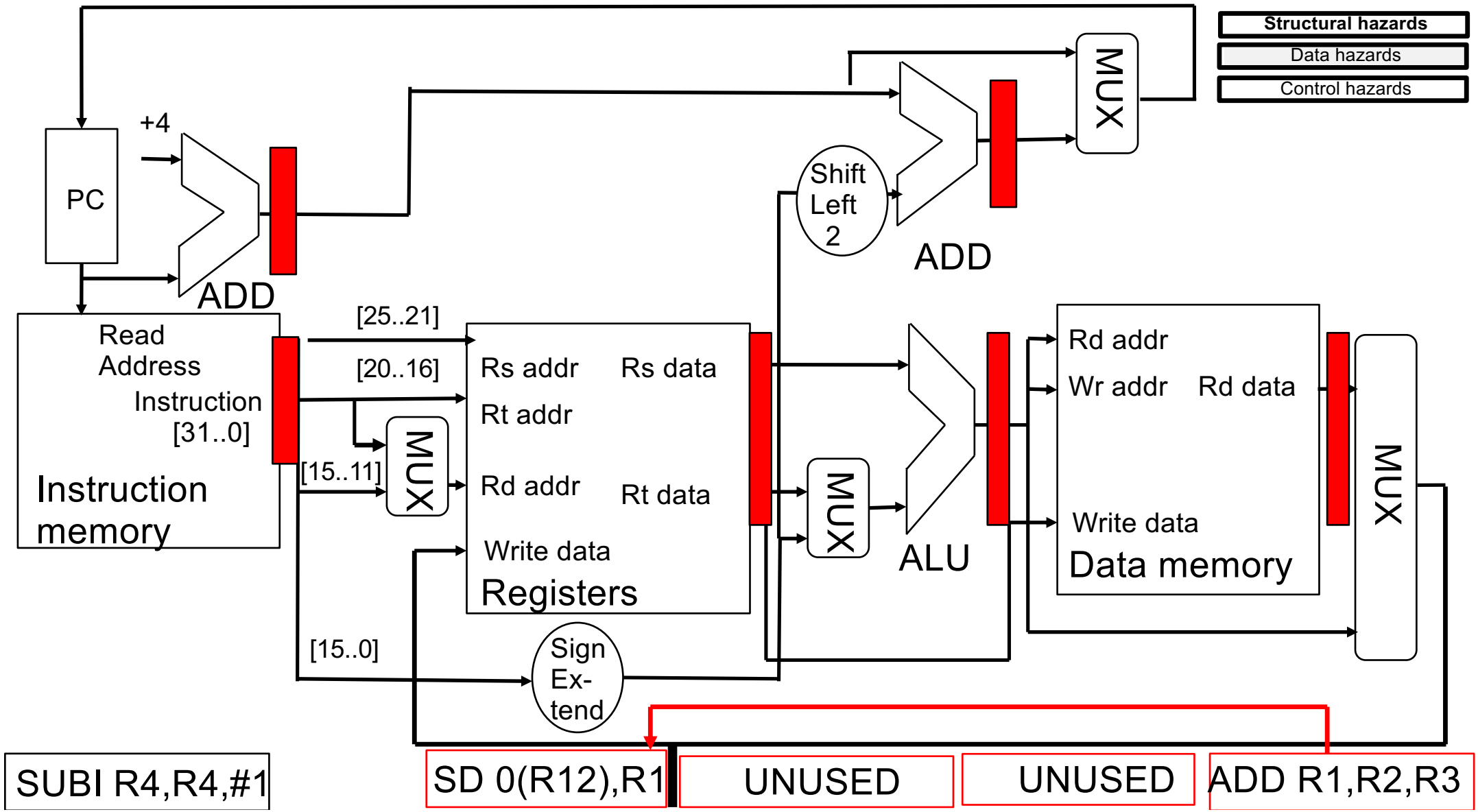

LABEL: LD R2, 0(R10)
LD R3, 0(R11)
ADD R1,R2,R3
SD 0(R12), R1
SUBI R4,R4,#1
ADDI R10,R10,#4
ADDI R11,R11,#4
ADDI R12,R12,#4
BNEZ R4, LABEL
PC
+4
ADD
MUX
Shift Left 2
ADD
Read Address
Instruction [31..0]
Instruction memory
[25..21]
[20..16]
[15..11]
MUX
Rs addr
Rs data
Rt addr
Rd addr
Rt data
Write data
Registers
[15..0]
Sign Ex-tend
MUX
ALU
Rd addr
Wr addr
Rd data
Write data
Data memory
MUX
INSTR 6
INSTR 5
INSTR 4
INSTR 3
INSTR 2

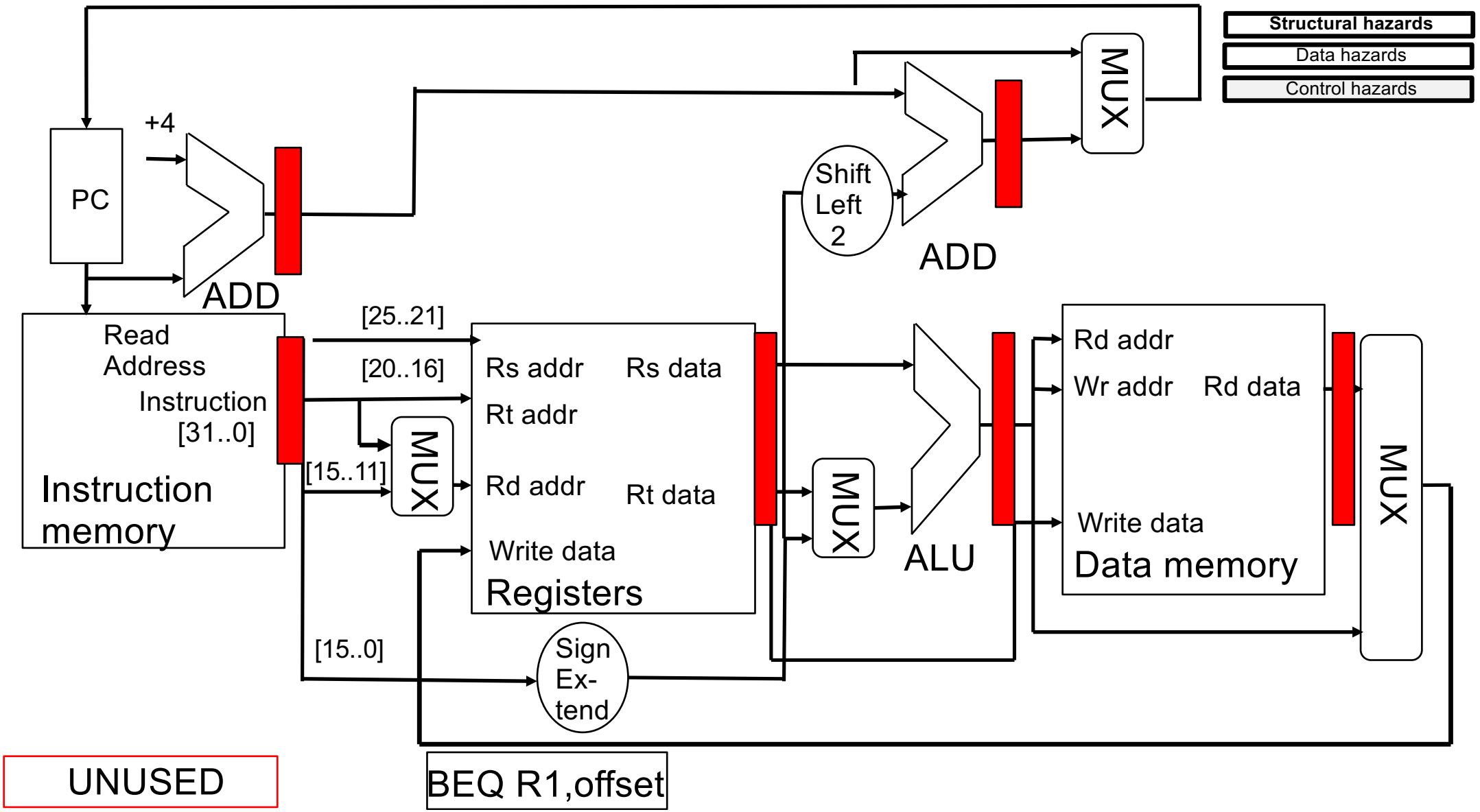

LABEL: LD R2, 0(R10)
LD R3, 0(R11)
ADD R1,R2,R3
SD 0(R12), R1
SUBI R4,R4,#1
ADDI R10,R10,#4
ADDI R11,R11,#4
ADDI R12,R12,#4
BNEZ R4, LABEL
PC
+4
ADD
MUX
Shift Left 2
ADD
Read Address
Instruction [31..0]
Instruction memory
[25..21]
[20..16]
[15..11]
MUX
Rs addr
Rs data
Rt addr
Rd addr
Rt data
Write data
Registers
[15..0]
Sign Ex-tend
MUX
ALU
Rd addr
Wr addr
Rd data
Write data
Data memory
MUX
INSTR 7
INSTR 6
INSTR 5
INSTR 4
INSTR 3

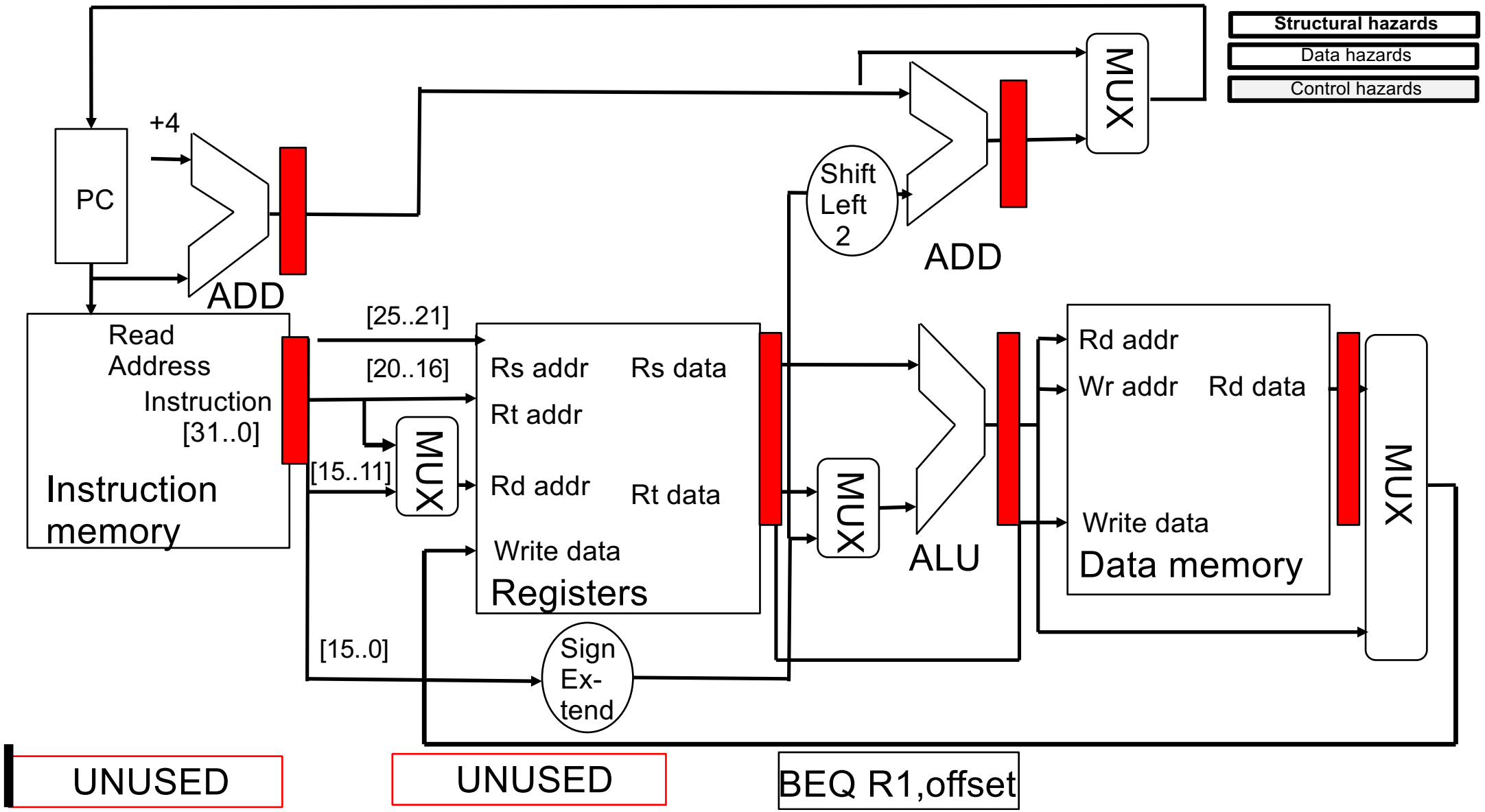

LABEL: LD R2, 0(R10)
LD R3, 0(R11)
ADD R1,R2,R3
SD 0(R12), R1
SUBI R4,R4,#1
ADDI R10,R10,#4
ADDI R11,R11,#4
ADDI R12,R12,#4
BNEZ R4, LABEL
PC
+4
ADD
MUX
Shift Left 2
ADD
Read Address
Instruction [31..0]
Instruction memory
[25..21]
[20..16]
[15..11]
[15..0]
MUX
Rs addr
Rs data
Rt addr
Rd addr
Rt data
Write data
Registers
Sign Ex-tend
MUX
ALU
Rd addr
Wr addr
Rd data
Write data
Data memory
MUX
SUBI R4,R4,#1
SD 0(R12),R1
ADD R1,R2,R3
LD R3,0(R11)
LD R2,0(R10)

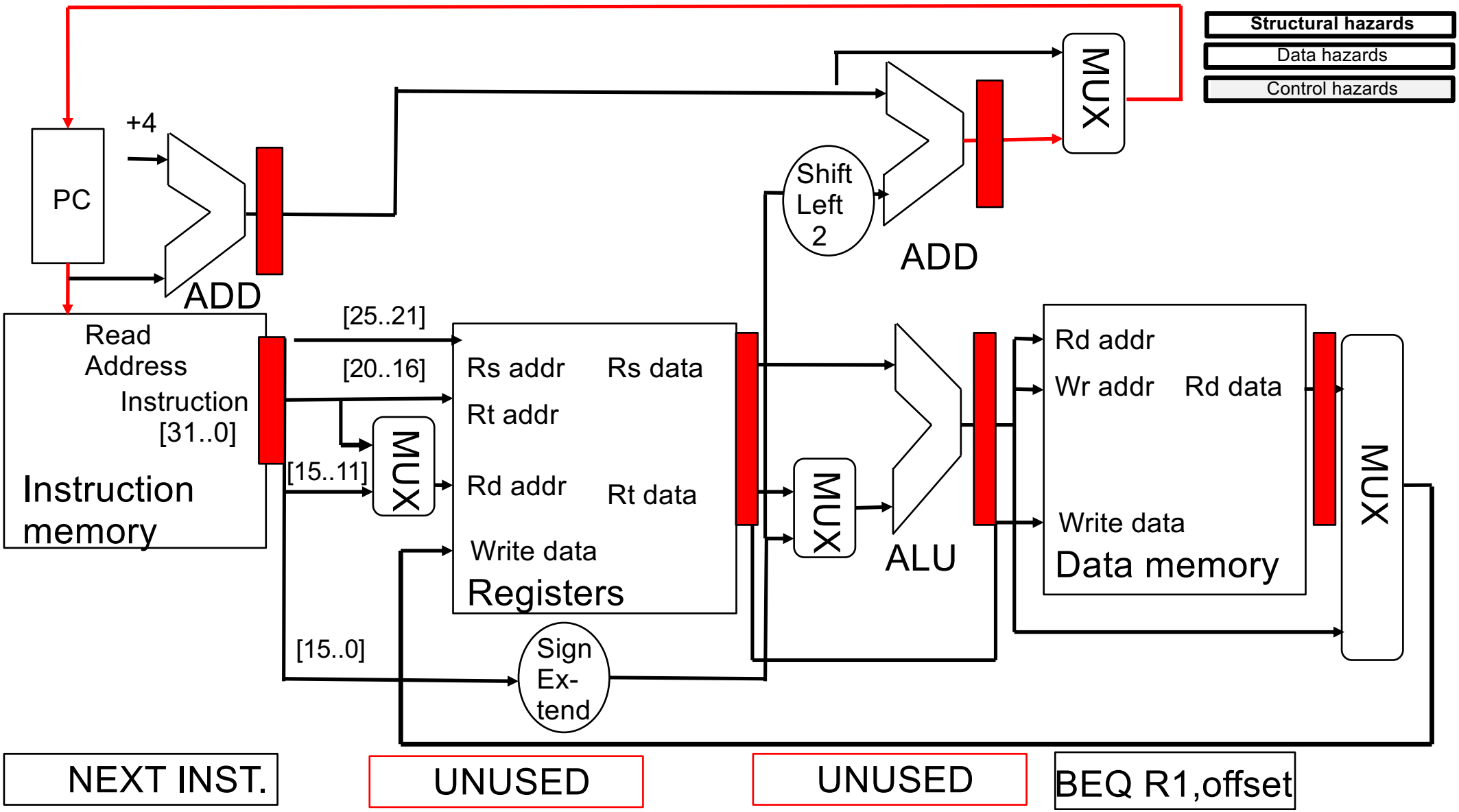

Structural hazards
Data hazards
Control hazards
PC
+4
ADD
MUX
Shift Left 2
ADD
Read Address
Instruction [31..0]
Instruction memory
[25..21]
[20..16]
[15..11]
MUX
Rs addr
Rs data
Rt addr
Rd addr
Rt data
Write data
Registers
MUX
ALU
Rd addr
Wr addr
Rd data
Write data
Data memory
MUX
[15..0]
Sign Ex-tend
SUBI R4,R4,#1
SD 0(R12),R1
ADD R1,R2,R3
LD R3,0(R11)
LD R2,0(R10)


Structural hazards
Data hazards
Control hazards
PC
+4
ADD
MUX
Shift Left 2
ADD
Read Address
Instruction [31..0]
Instruction memory
[25..21]
[20..16]
[15..11]
[15..0]
MUX
Rs addr
Rs data
Rt addr
Rd addr
Rt data
Write data
Registers
Sign Ex-tend
MUX
ALU
Rd addr
Wr addr
Rd data
Write data
Data memory
MUX
SUBI R4,R4,#1
SD 0(R12),R1
UNUSED
ADD R1,R2,R3
LD R3,0(R11)

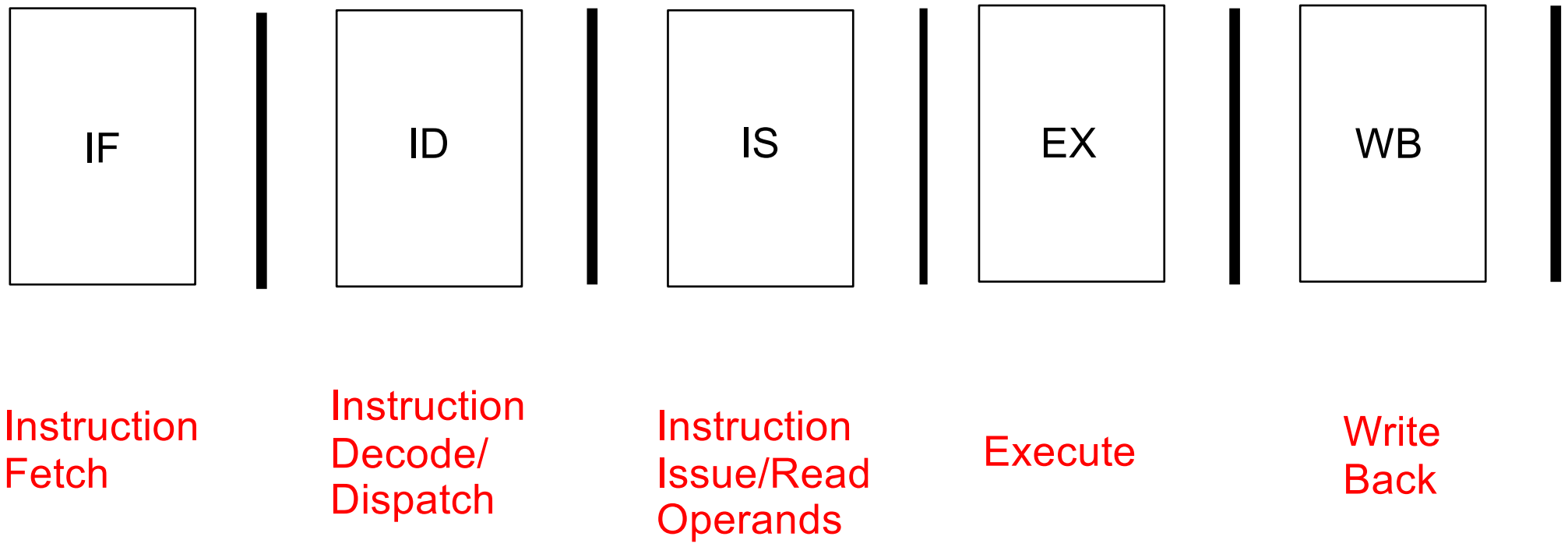

Structural hazards
Data hazards
Control hazards
PC
+4
ADD
MUX
Shift Left 2
ADD
Read Address
Instruction [31..0]
Instruction memory
[25..21]
[20..16]
[15..11]
[15..0]
MUX
Rs addr
Rs data
Rt addr
Rd addr
Rt data
Write data
Registers
Sign Ex-tend
MUX
ALU
Rd addr
Wr addr
Rd data
Write data
Data memory
MUX
SUBI R4,R4,#1
SD 0(R12),R1
UNUSED
UNUSED
ADD R1,R2,R3

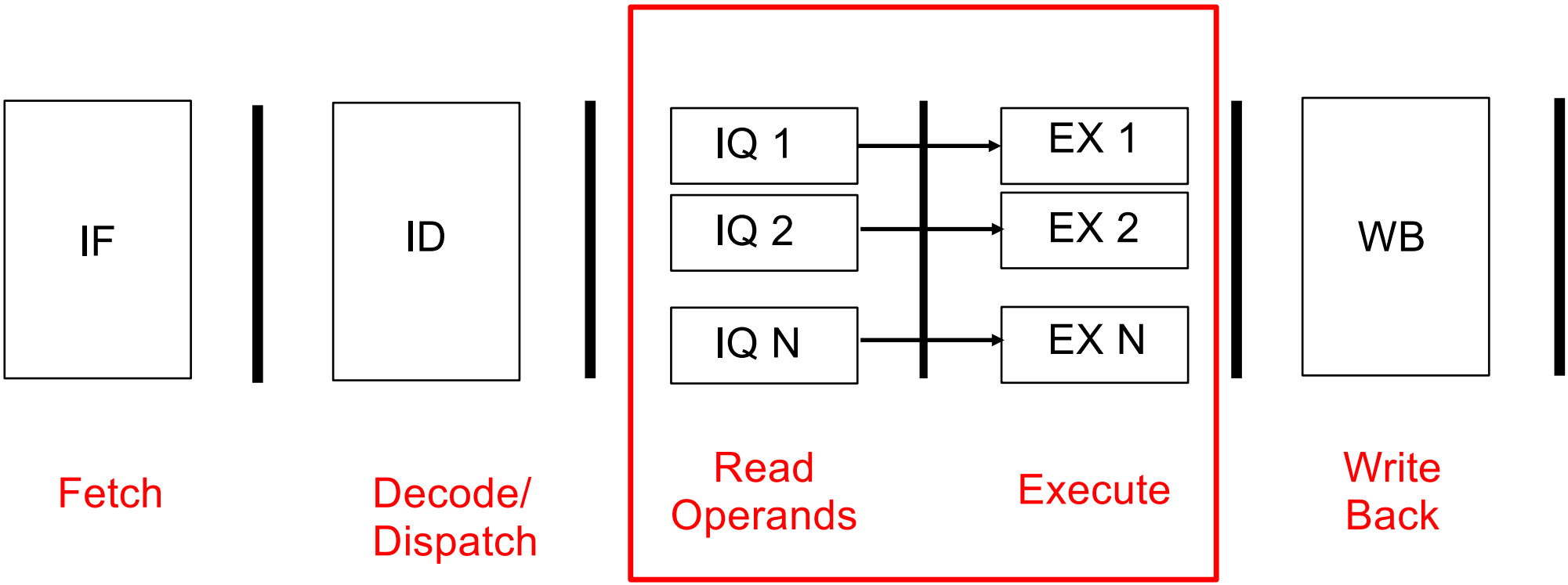

Structural hazards
Data hazards
Control hazards
PC
+4
ADD
MUX
Shift
Left
2
ADD
Read
Address
Instruction
[31..0]
Instruction
memory
[25..21]
[20..16]
[15..11]
MUX
Rs addr
Rs data
Rt addr
Rd addr
Rt data
Write data
Registers
MUX
ALU
Rd addr
Wr addr
Rd data
Write data
Data memory
MUX
[15..0]
Sign
Ex-
tend
UNUSED
BEQ R1,offset

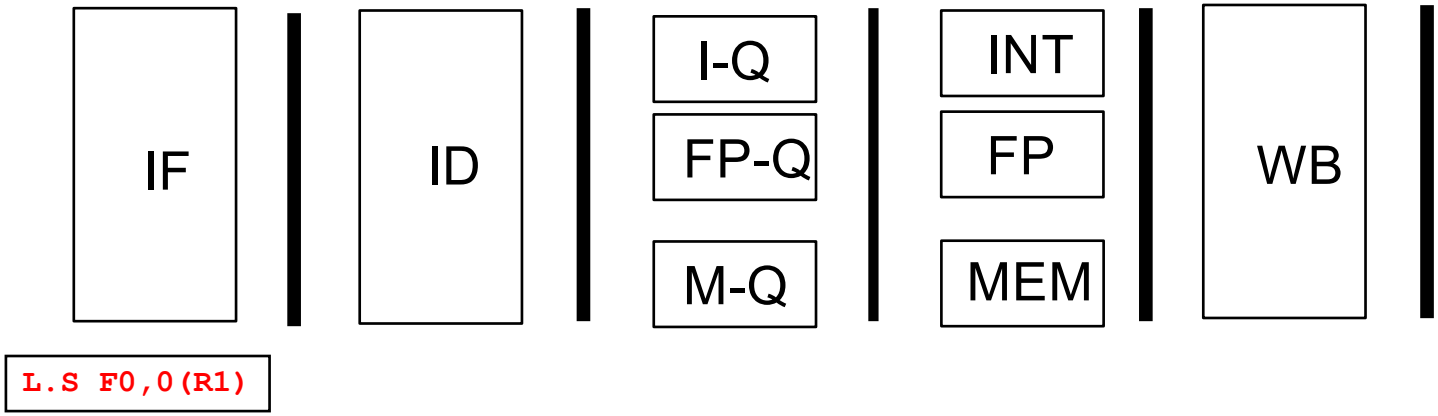

Structural hazards
Data hazards
Control hazards
PC
+4
ADD
MUX
Shift
Left
2
ADD
Read
Address
Instruction
[31..0]
Instruction
memory
[25..21]
[20..16]
[15..11]
MUX
Rs addr
Rs data
Rt addr
Rd addr
Rt data
Write data
Registers
MUX
ALU
Rd addr
Wr addr
Rd data
Write data
Data memory
MUX
[15..0]
Sign
Ex-
tend
UNUSED
UNUSED
BEQ R1,offset


Structural hazards
Data hazards
Control hazards
PC
+4
ADD
MUX
Shift Left 2
ADD
Read Address
Instruction [31..0]
Instruction memory
[25..21]
[20..16]
[15..11]
MUX
Rs addr
Rs data
Rt addr
Rd addr
Rt data
Write data
Registers
MUX
ALU
Rd addr
Wr addr
Rd data
Write data
Data memory
MUX
[15..0]
Sign Ex-tend
NEXT INST.
UNUSED
UNUSED
BEQ R1,offset

Superscalar Microprocessors

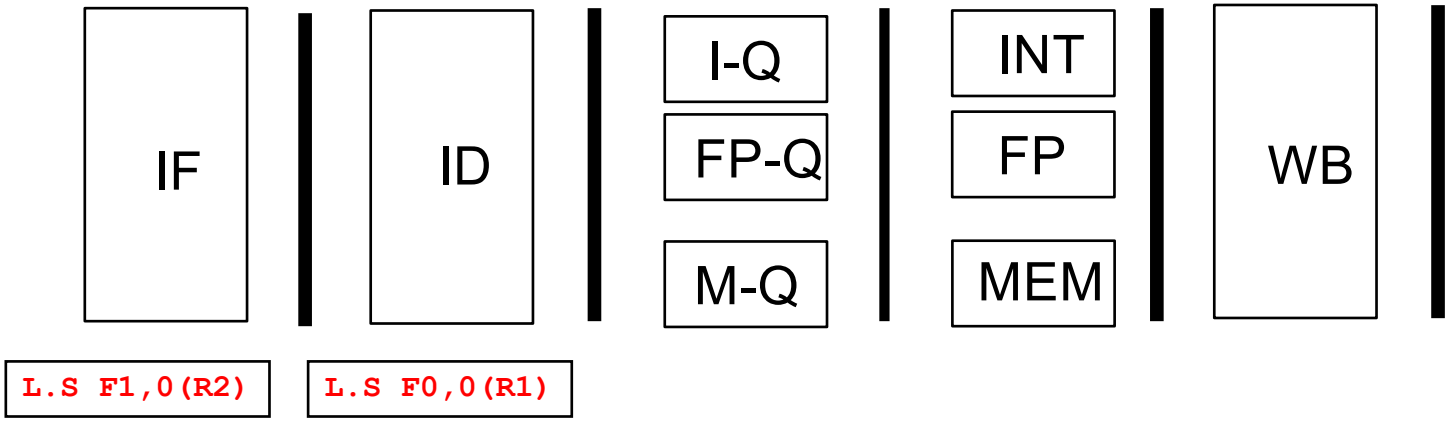

IF
ID
IS
EX
WB
Instruction Fetch
Instruction Decode/ Dispatch
Instruction Issue/Read Operands
Execute
Write Back


IF
ID
IQ 1
IQ 2
IQ N
EX 1
EX 2
EX N
WB
Fetch
Decode/
Dispatch
Read
Operands
Execute
Write
Back



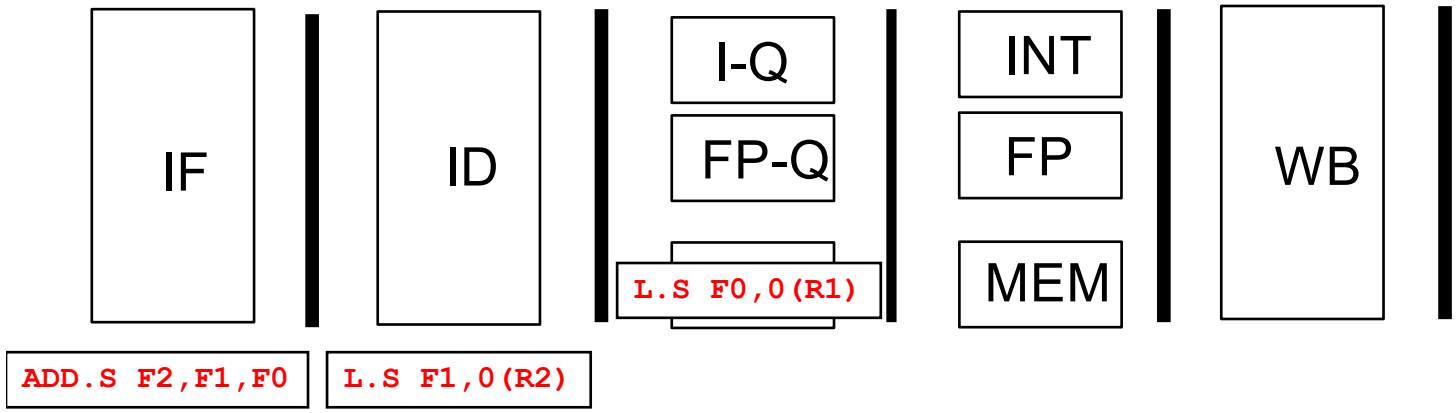
Cycle: 1

```
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
```



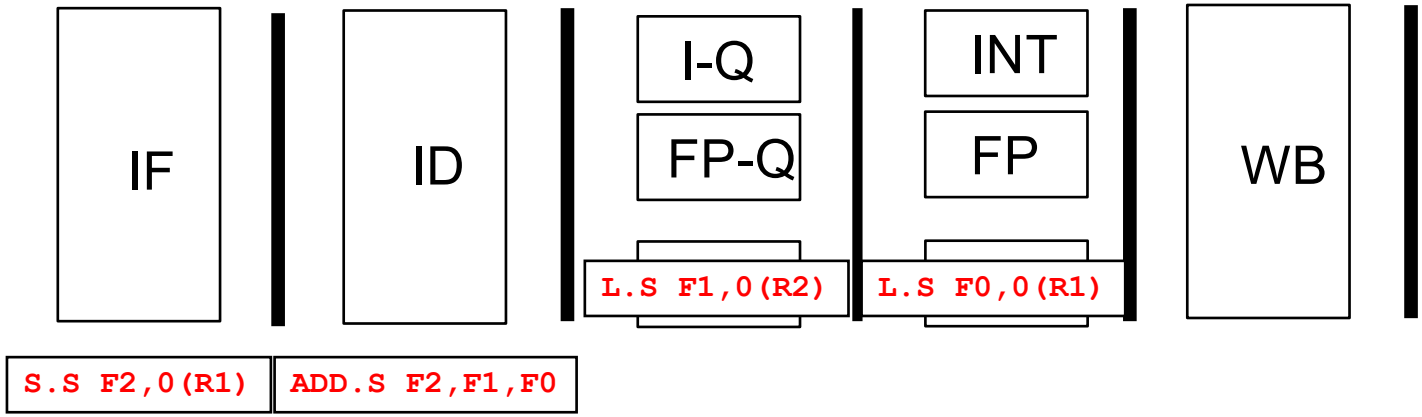
Cycle: 2

```
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
```



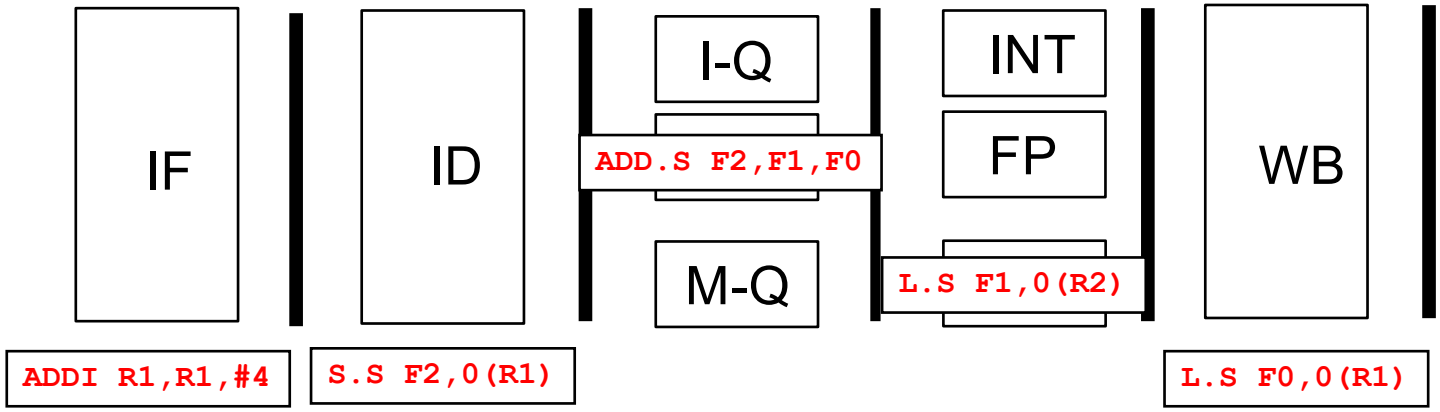
Cycle: 3

```
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
```



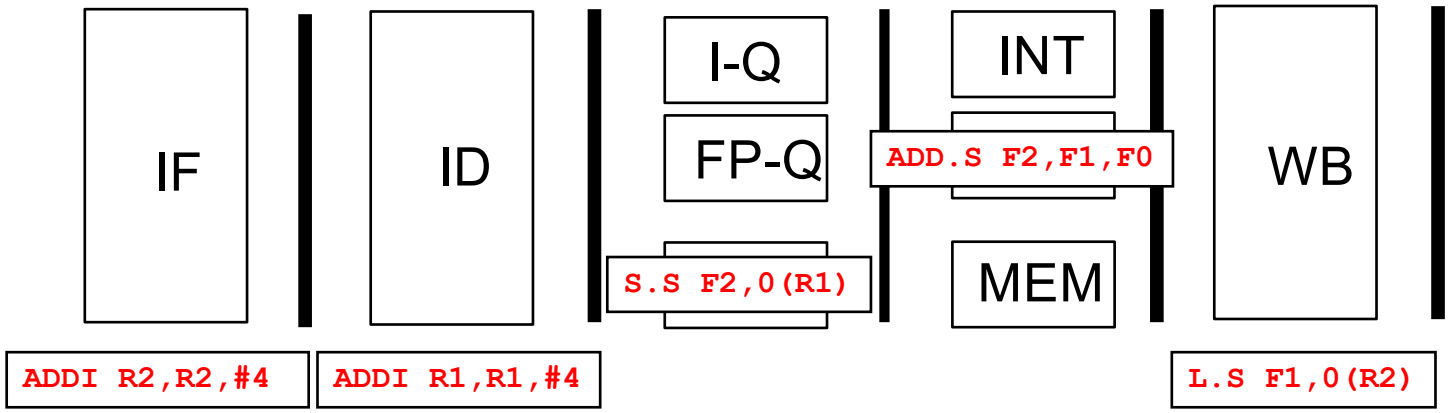
Cycle: 4

```
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
```



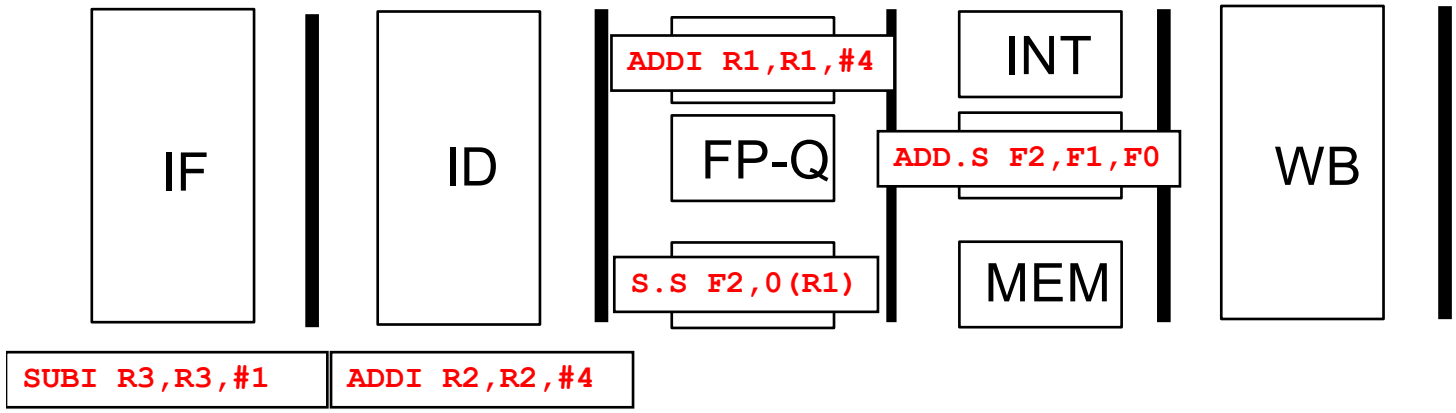
Cycle: 5

```
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
```



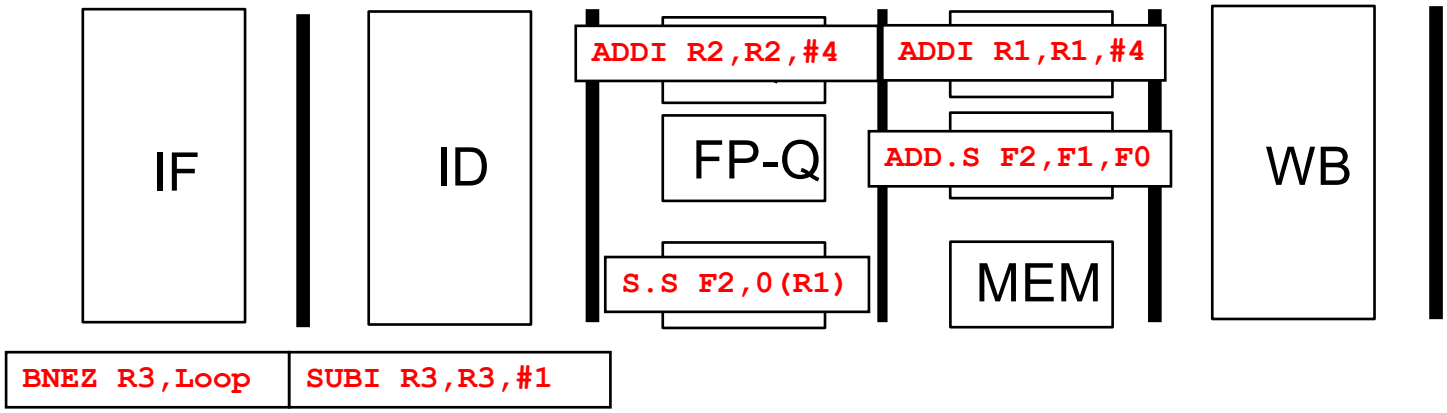
Cycle: 6

```
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

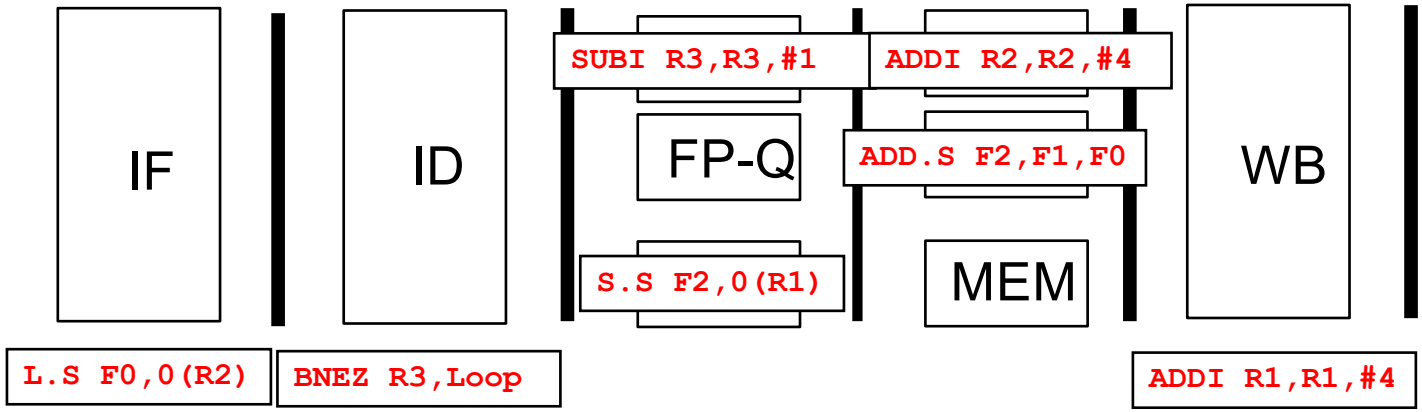
```
Loop    L.S  F0,0(R1)
        L.S  F1,0(R2)
        ADD.S F2,F1,F0
        S.S  F2,0(R1)
        ADDI R1,R1,#4
        ADDI R2,R2,#4
        SUBI R3,R3,#1
        BNEZ R3,Loop
```



Cycle: 8

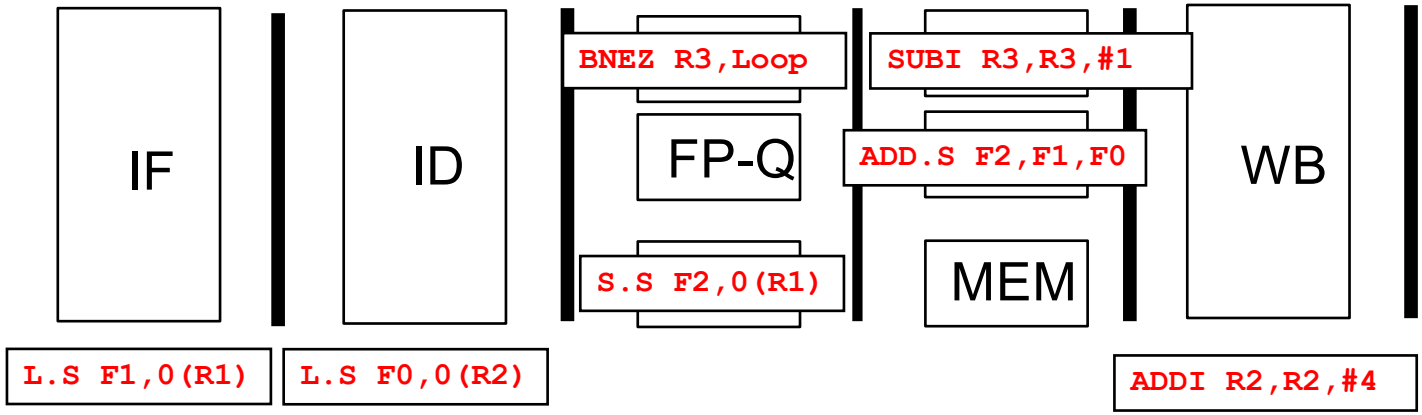
The ADDI R1,R1,#4 is being executed Out-of-program-order with respect to the S.S F2,0(R1)!

```
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
```



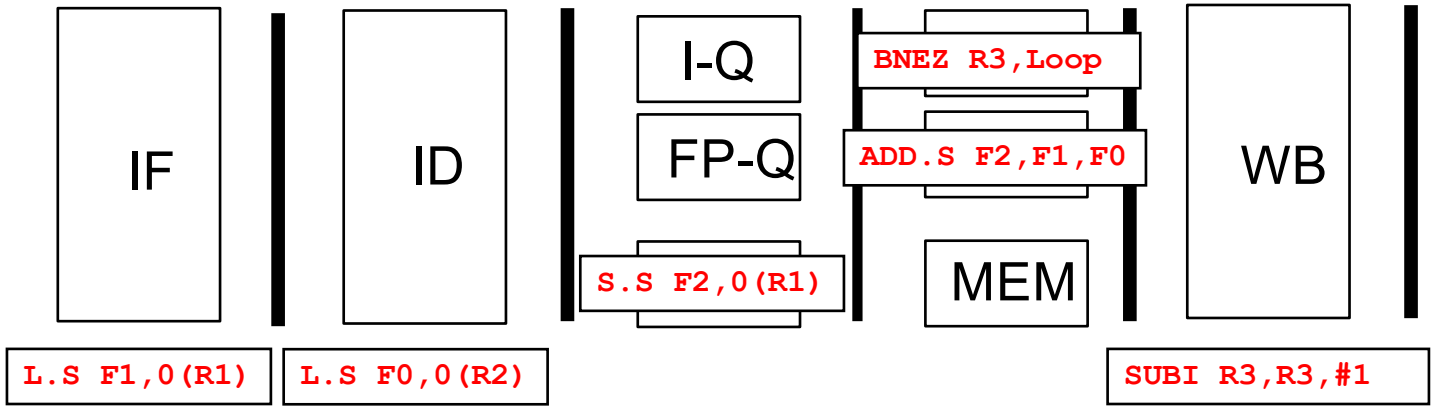
Cycle: 9

```
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
```



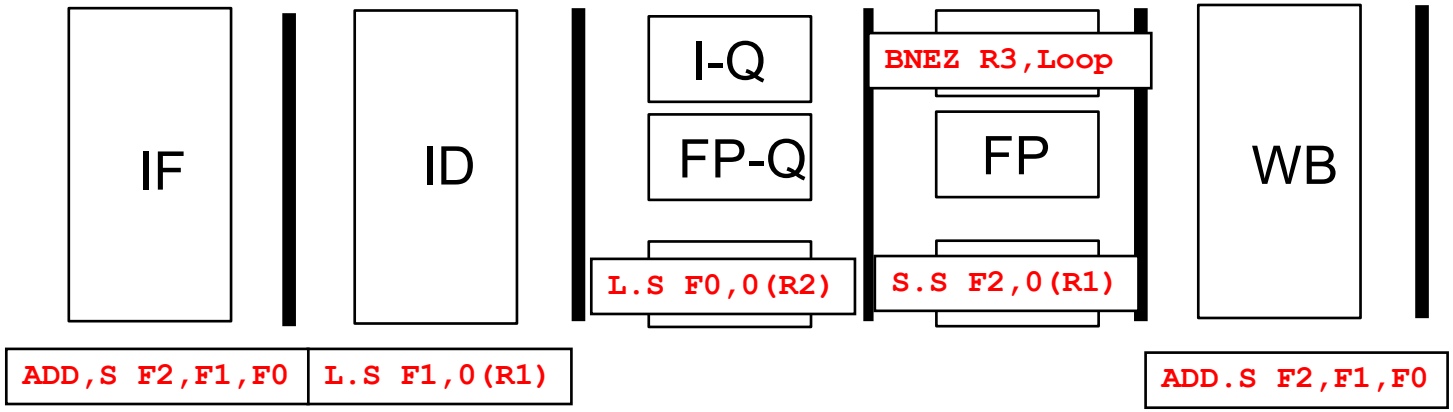
Cycle: 10

```
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

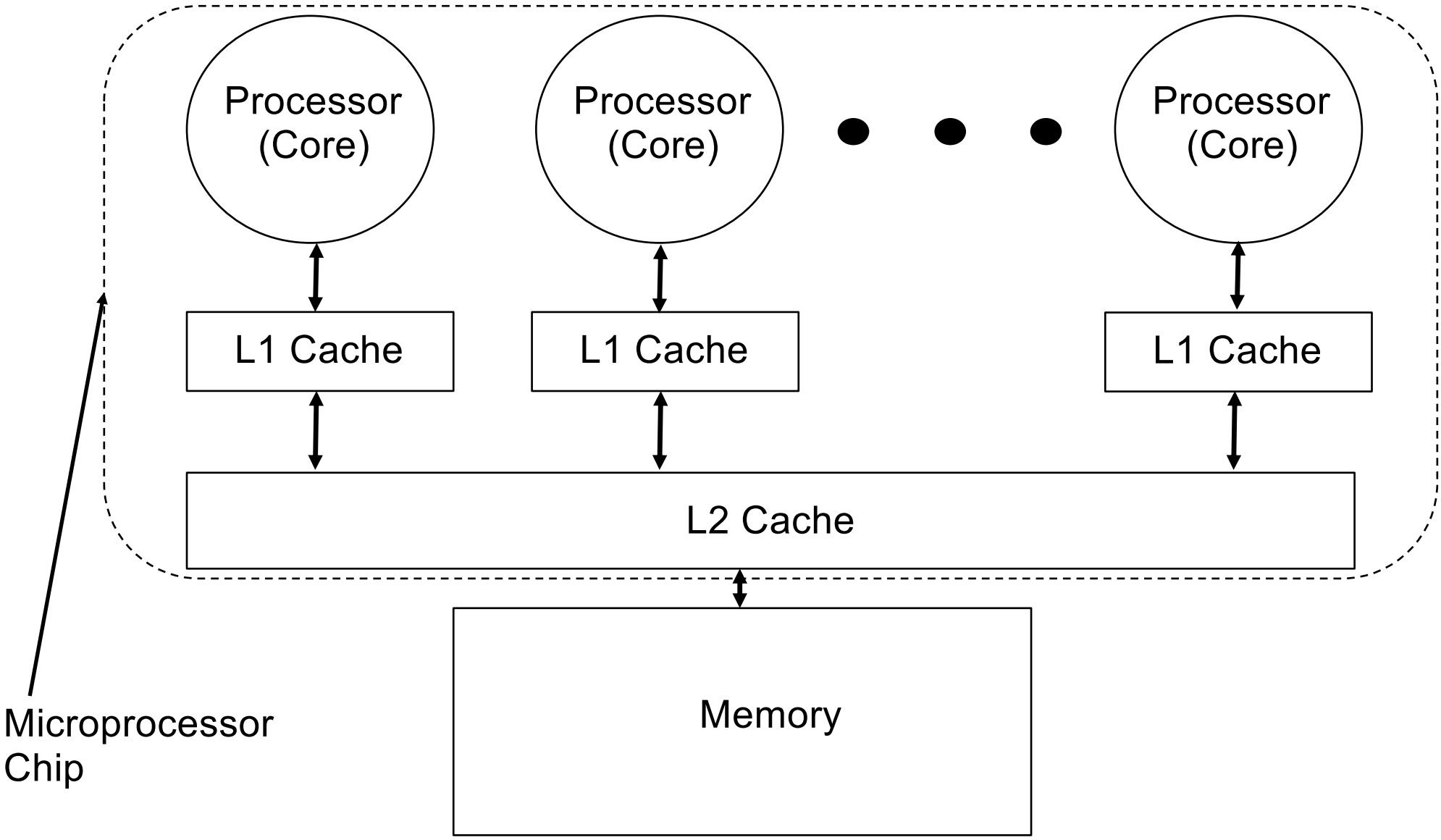
```
Loop    L.S  F0,0(R1)
        L.S  F1,0(R2)
        ADD.S F2,F1,F0
        S.S  F2,0(R1)
        ADDI R1,R1,#4
        ADDI R2,R2,#4
        SUBI R3,R3,#1
        BNEZ R3,Loop
```



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
```

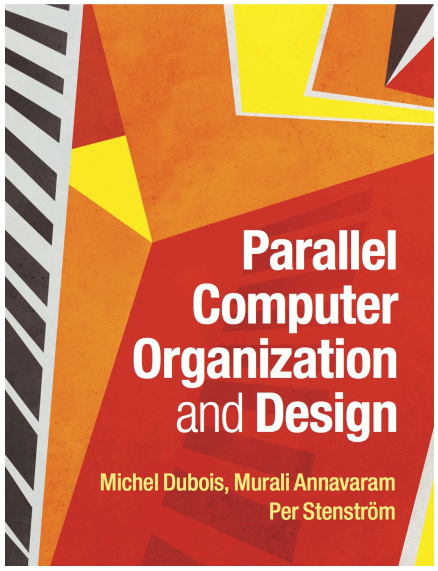
Multicore Computers


Processor
(Core)
Processor
(Core)
Processor
(Core)
L1 Cache
L1 Cache
L1 Cache
L2 Cache
Memory
Microprocessor
Chip

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



- Two MOOCs developed at Chalmers
  - *Computer Systems Design for Energy Efficiency*
  - *Computer Systems Design: Advanced Concepts of Modern Microprocessors*

- *Hosted on EdX platform*
- *Targets both performance and energy-efficiency*
- *February – March 2017*
- *> 2000 students*