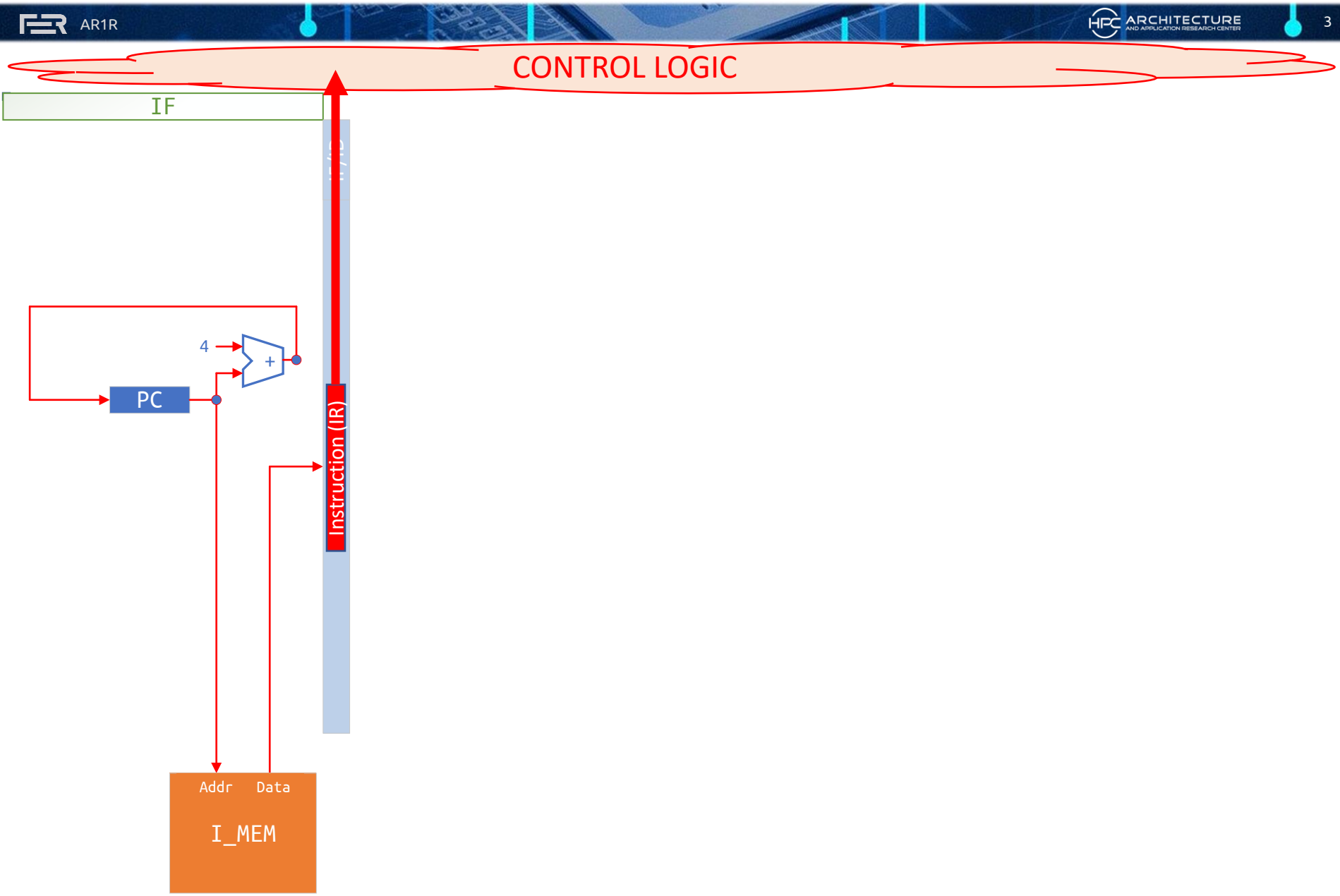
FRISC-V

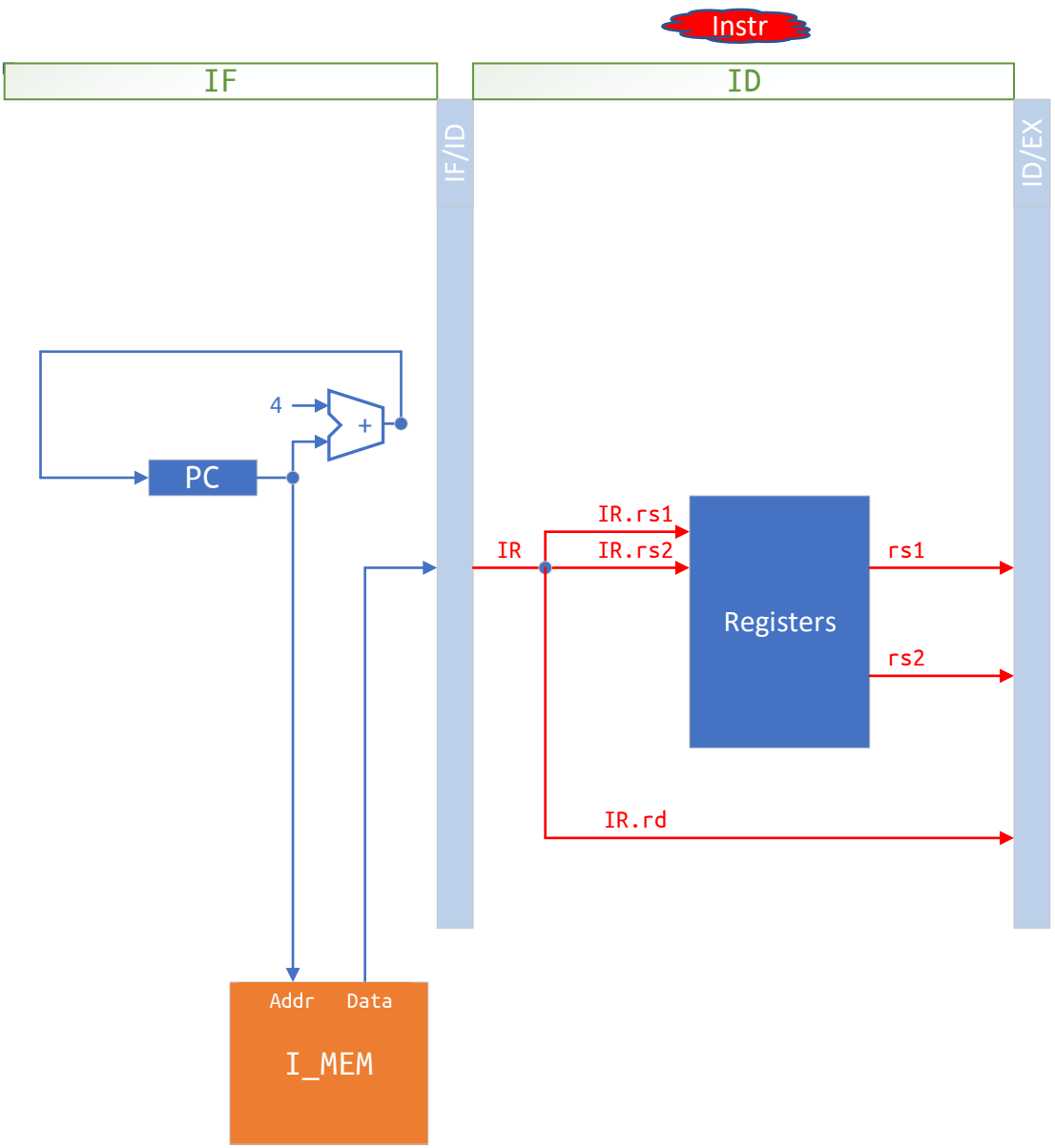
Mikroarhitektura puta podataka

Put podataka v1.0

- Osnovne AL naredbe
  - *Operacija* rd,rs1,rs2

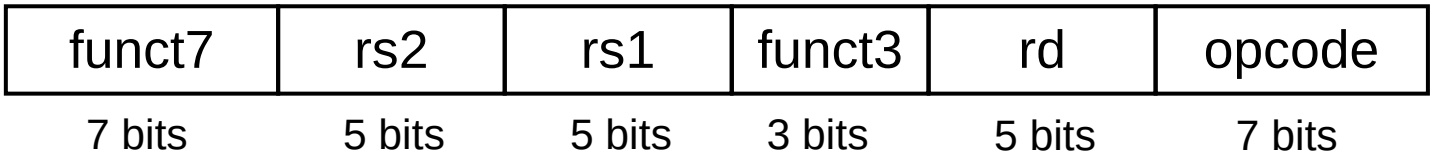
- add, sub, and, or, xor, sll, srl, sra, slt, sltu


CONTROL LOGIC
IF
I/D
4
+
PC
Instruction (IR)
Addr
Data
I_MEM

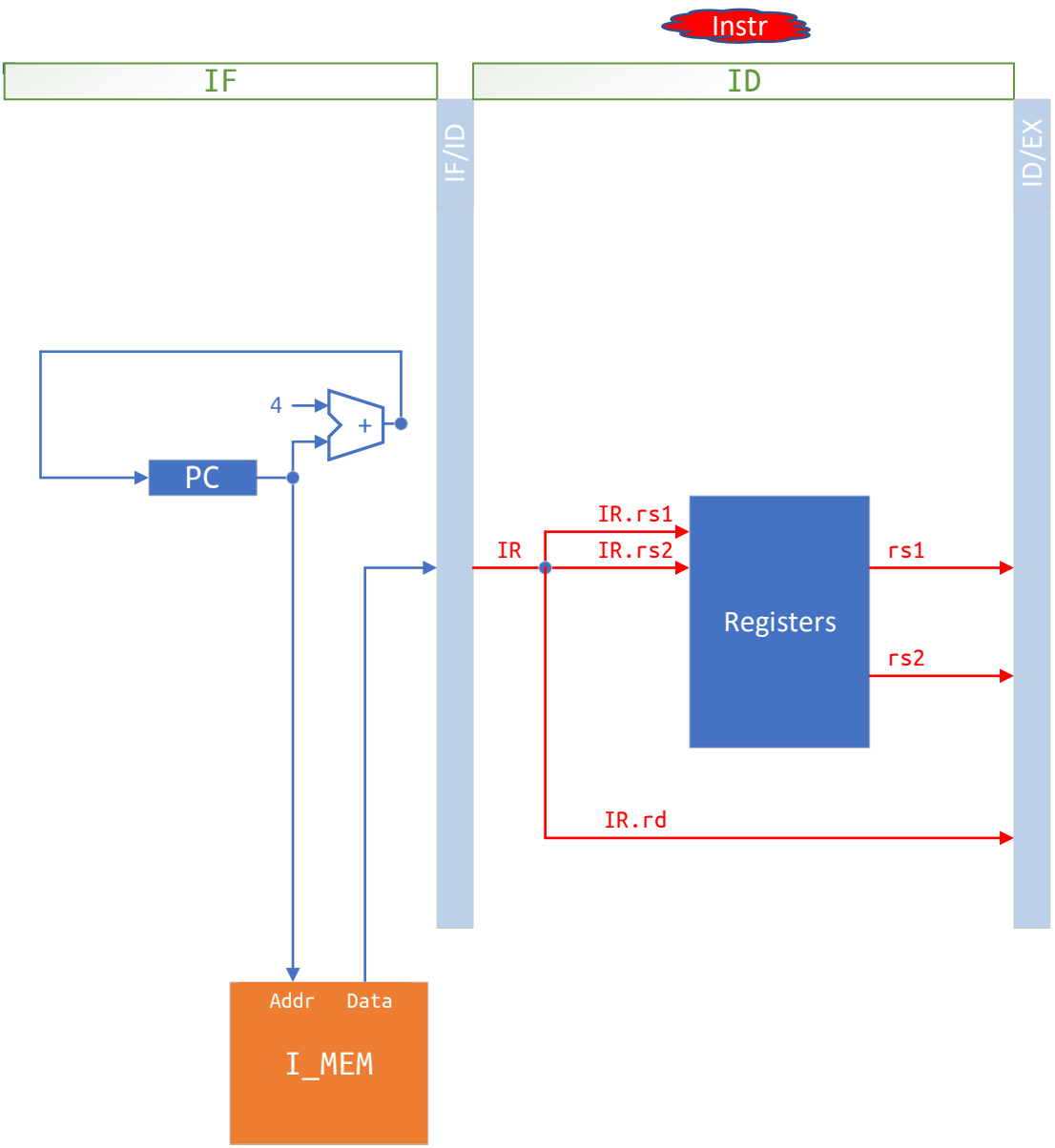

Instr
IF
ID
IF/ID
ID/EX
4
+
PC
IR.rs1
IR.rs2
IR
Registers
rs1
rs2
IR.rd
Addr
Data
I_MEM

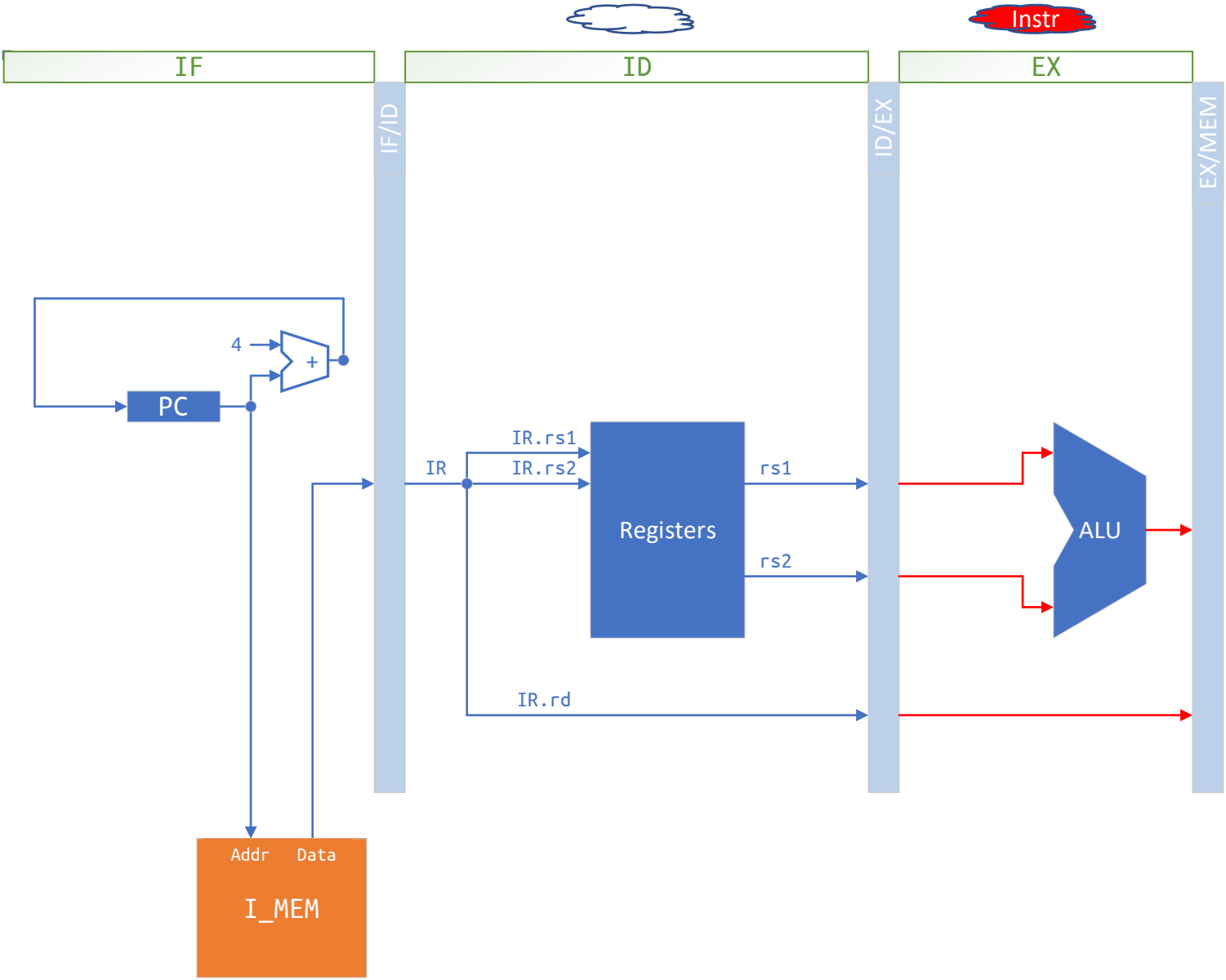
RISC-V R-format Instructions

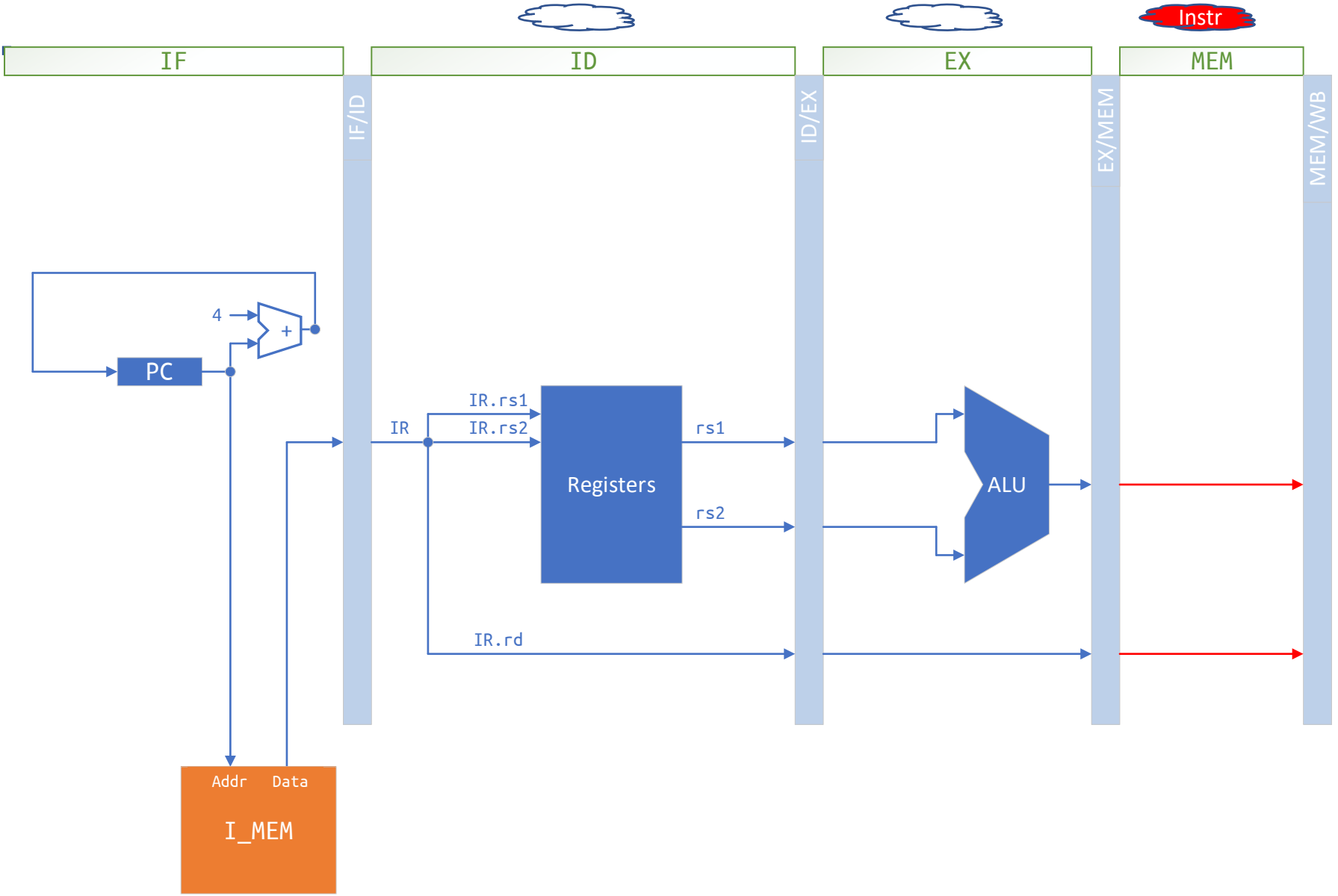
- Kako procesor zna koje registre izabrati ?

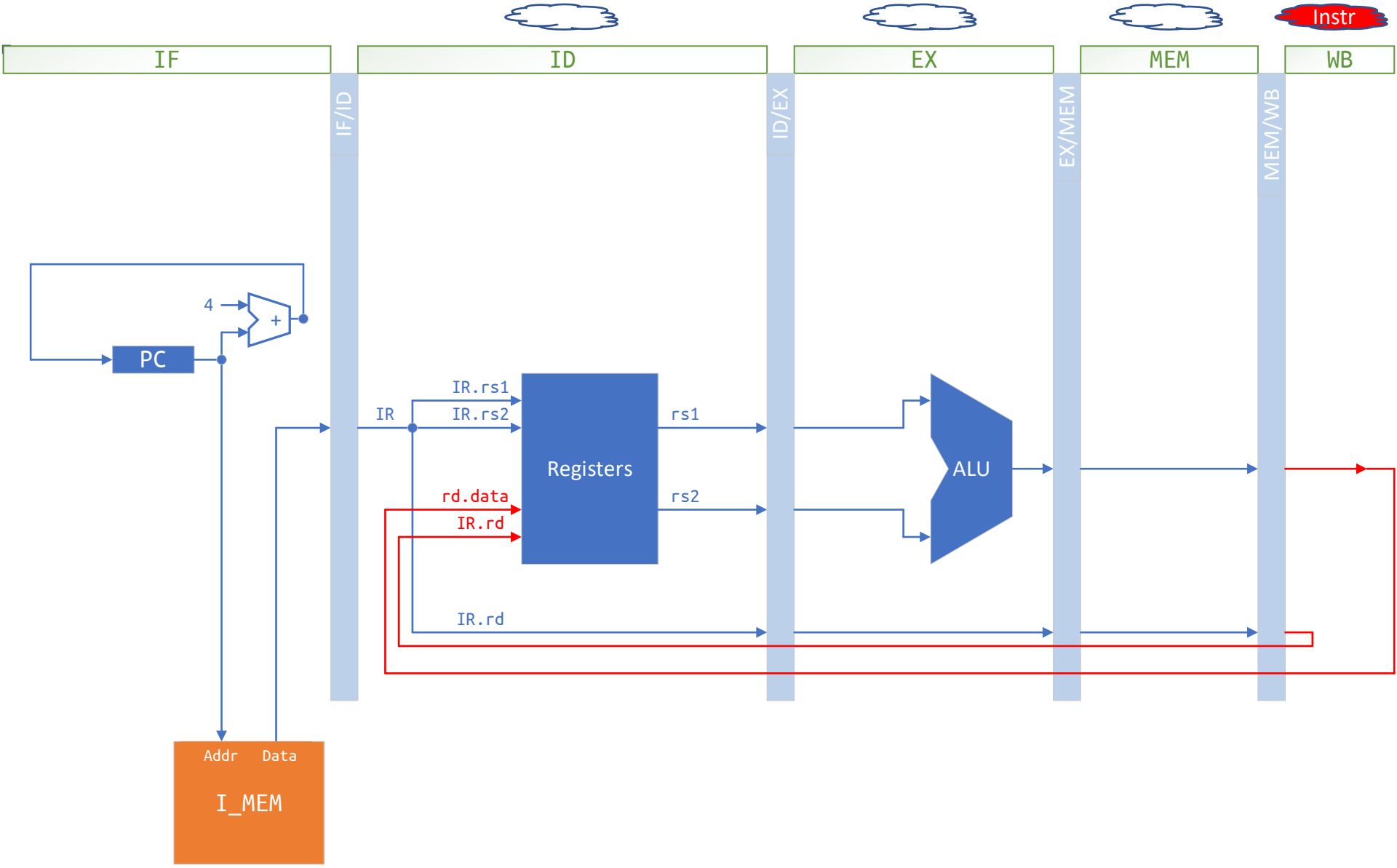


- Instruction fields
  - opcode: operation code
  - rd: destination register number
  - funct3: 3-bit function code (additional opcode)
  - rs1: the first source register number
  - rs2: the second source register number
  - funct7: 7-bit function code (additional opcode)


Instr
IF
ID
IF/ID
ID/EX
4
+
PC
IR.rs1
IR.rs2
IR
Registers
rs1
rs2
IR.rd
Addr
Data
I_MEM

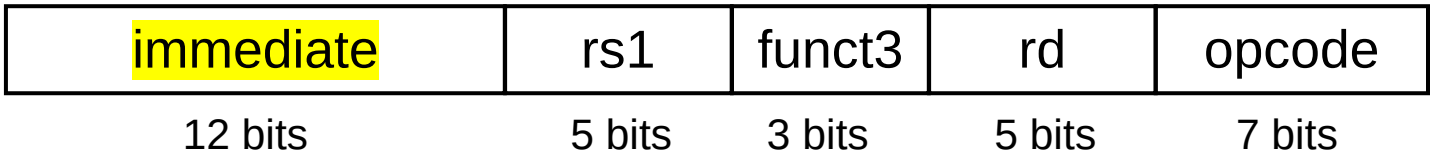

Instr
IF
ID
EX
IF/ID
ID/EX
EX/MEM
4
+
PC
IR
IR.rs1
IR.rs2
Registers
rs1
rs2
ALU
IR.rd
Addr
Data
I_MEM


Instr
IF
ID
EX
MEM
IF/ID
ID/EX
EX/MEM
MEM/WB
4
+
PC
IR
IR.rs1
IR.rs2
Registers
rs1
rs2
ALU
IR.rd
Addr
Data
I_MEM


Instr
IF
ID
EX
MEM
WB
IF/ID
ID/EX
EX/MEM
MEM/WB
4
+
PC
IR.rs1
IR
IR.rs2
Registers
rs1
rs2
rd.data
IR.rd
ALU
IR.rd
Addr
Data
I_MEM

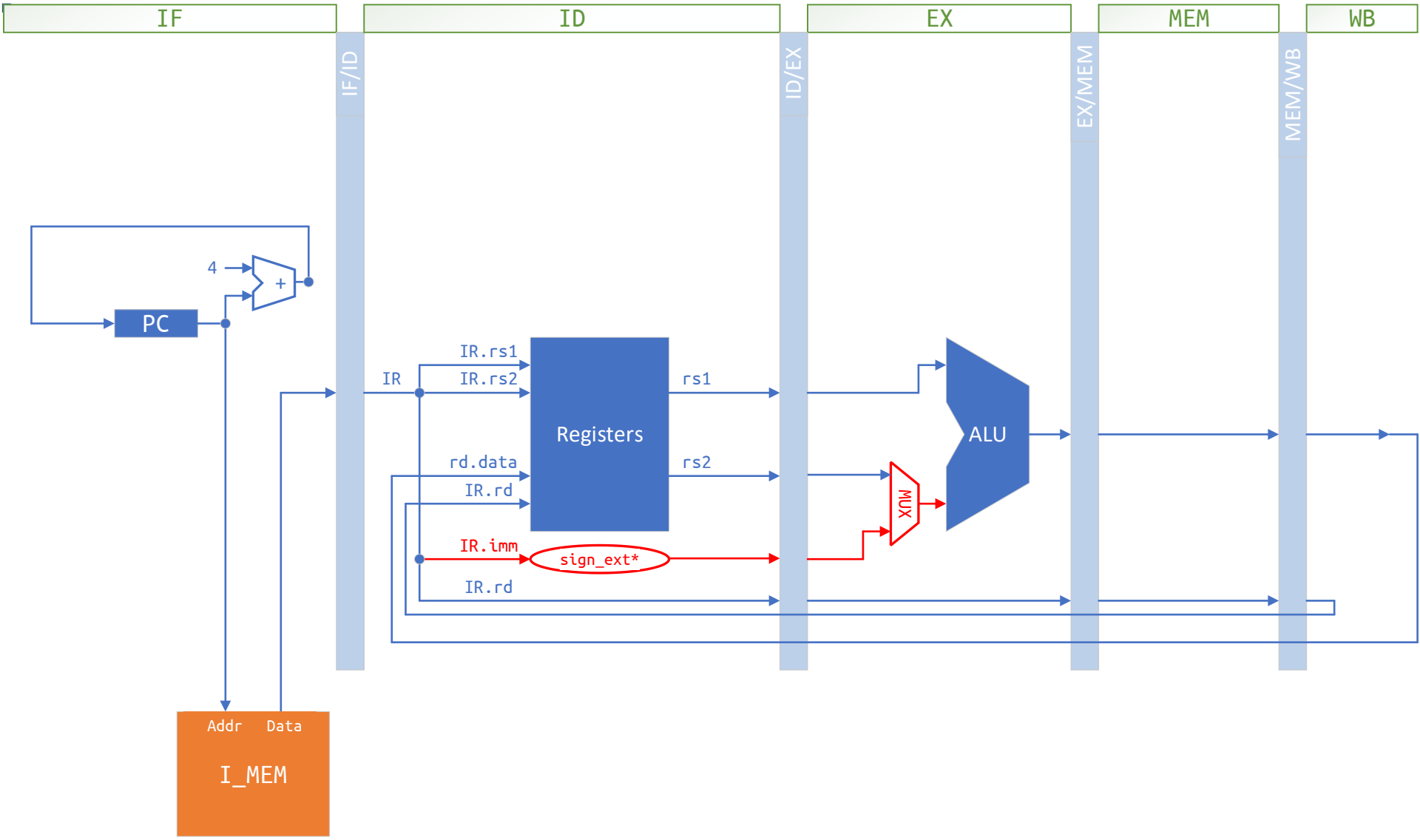
Put podataka v1.0

- Drugi operand može biti i neposredna vrijednost
  - addi, andi, ori, xori, slli, srli, srai, slti,sltiu



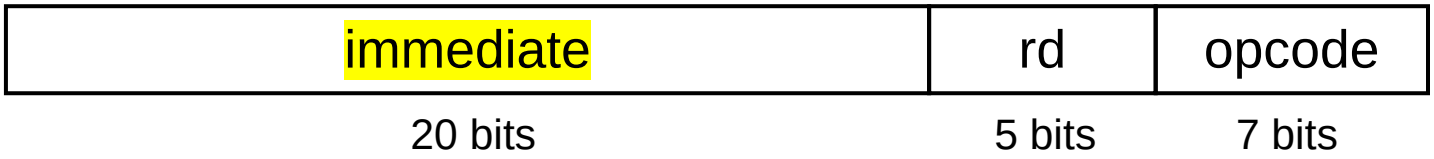
- RISC-V I-format Instructions
  - Immediate arithmetic
    - rs1: source
    - immediate: constant operand
      - 2s-complement, sign extended

- Kako ovo implementirati u našem putu podataka ?


IF
ID
EX
MEM
WB
IF/ID
ID/EX
EX/MEM
MEM/WB
4
+
PC
IR
IR.rs1
IR.rs2
Registers
rs1
rs2
rd.data
IR.rd
IR.imm
sign_ext*
IR.rd
ALU
MUX
Addr
Data
I_MEM

Put podataka v1.0

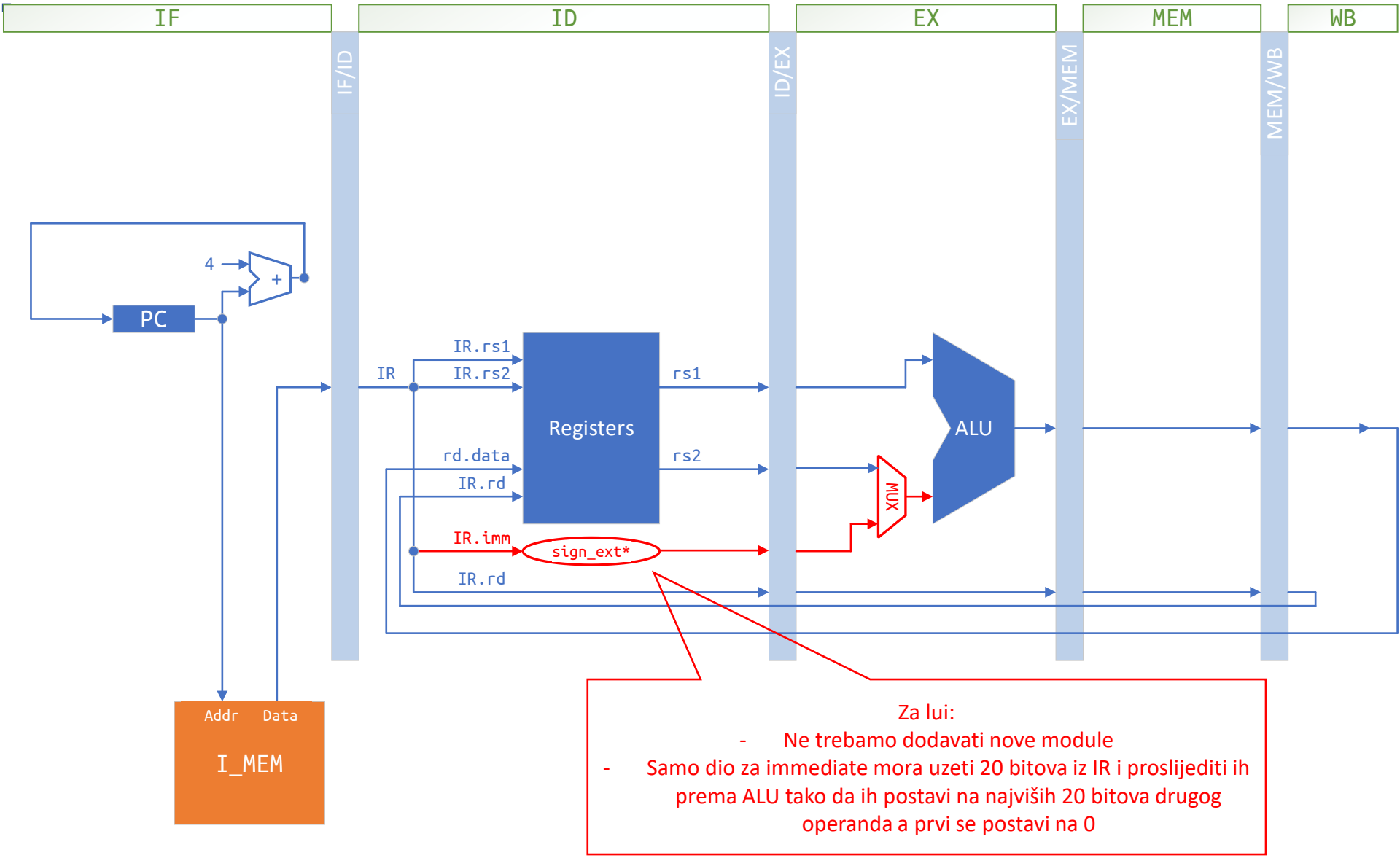
- Posebne AL naredbe:
  - lui
  - auipc



- RISC-V U-format Instructions

- Kako ovo implementirati u našem putu podataka ?

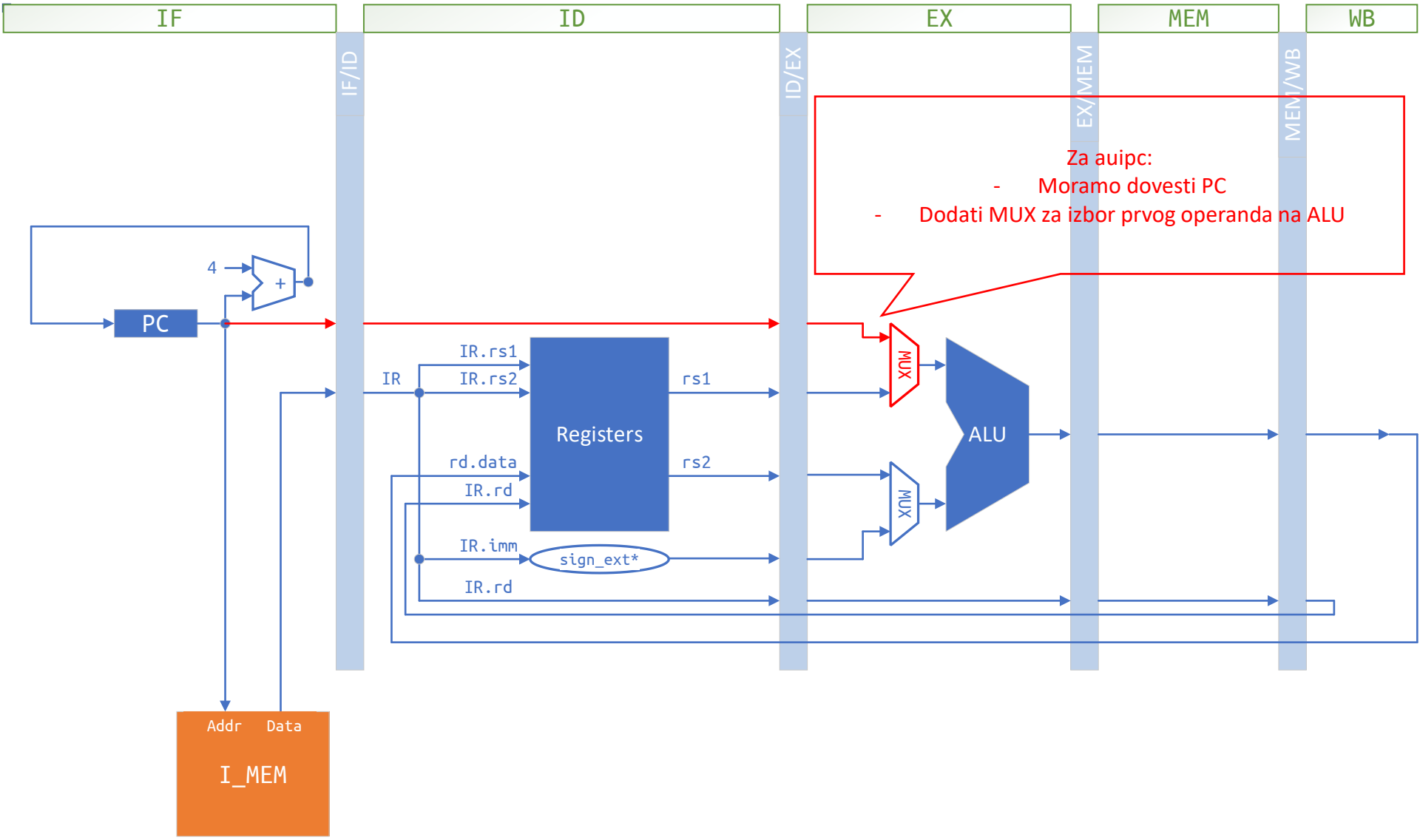
lui



Za lui:

- Ne trebamo dodavati nove module
- Samo dio za immediate mora uzeti 20 bitova iz IR i proslijediti ih prema ALU tako da ih postavi na najviših 20 bitova drugog operanda a prvi se postavi na 0

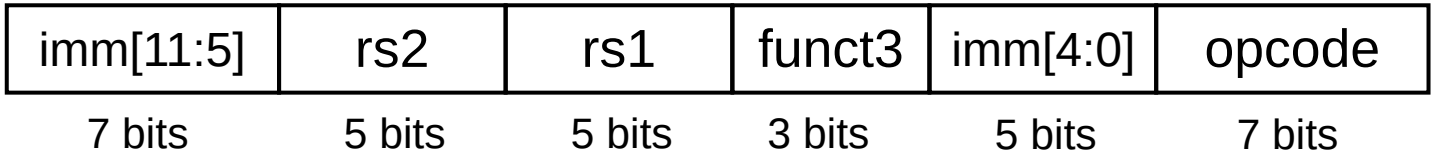
auipc


IF
ID
EX
MEM
WB
IF/ID
ID/EX
EX/MEM
MEM/WB
Za auipc:
- Moramo dovesti PC
- Dodati MUX za izbor prvog operanda na ALU
4
+
PC
IR
IR.rs1
IR.rs2
Registers
rs1
rs2
rd.data
IR.rd
IR.imm
sign_ext*
IR.rd
MUX
MUX
ALU
Addr
Data
I_MEM

Put podataka v1.0

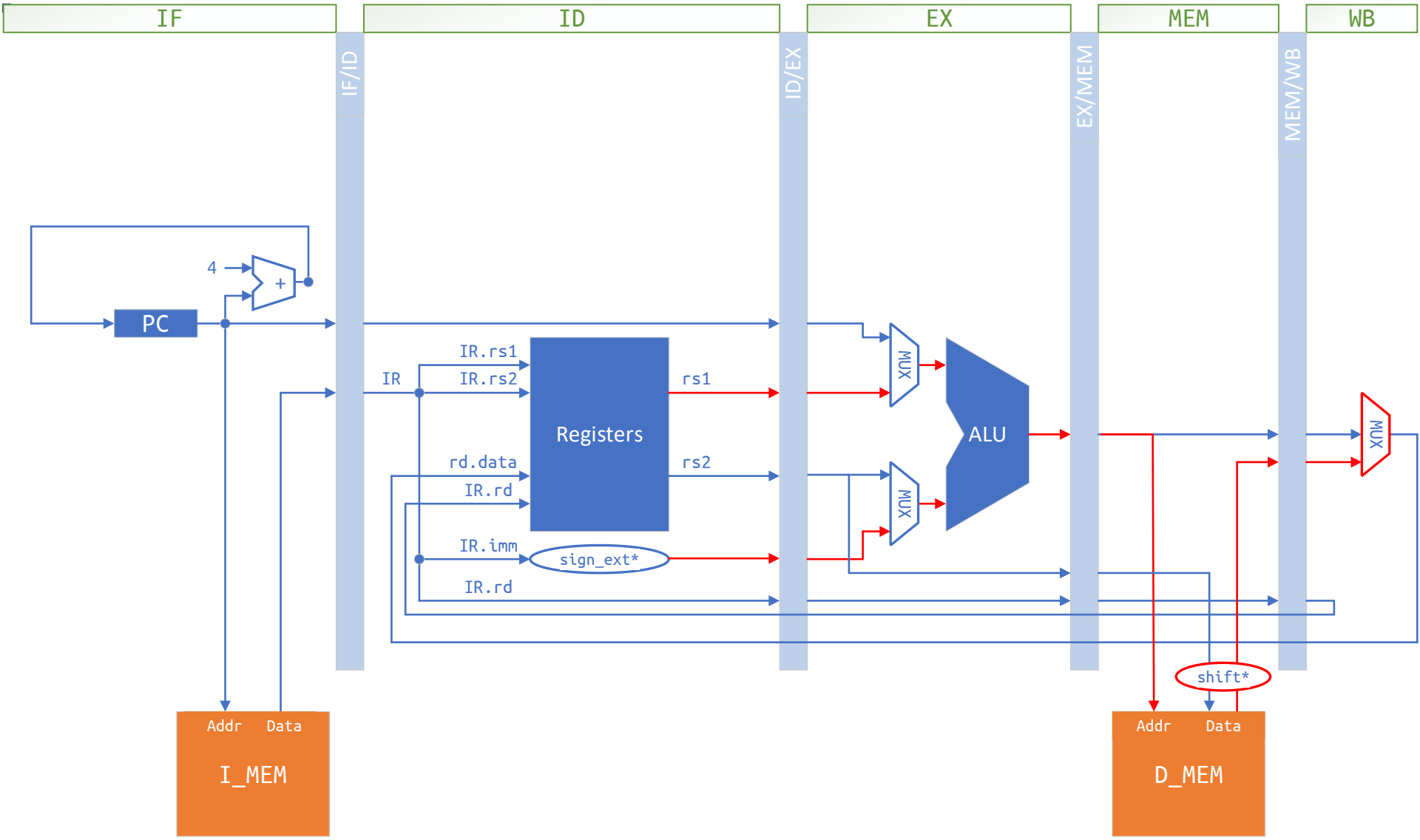
- Load i store naredbe
  - Load koristi isti format kao i AL naredbe sa immediate
    - Izračun adrese je u stvari zbrajanje rs1+imm
    - Rd za spremanje učitane vrijednost

  - Store koristi poseban S-format

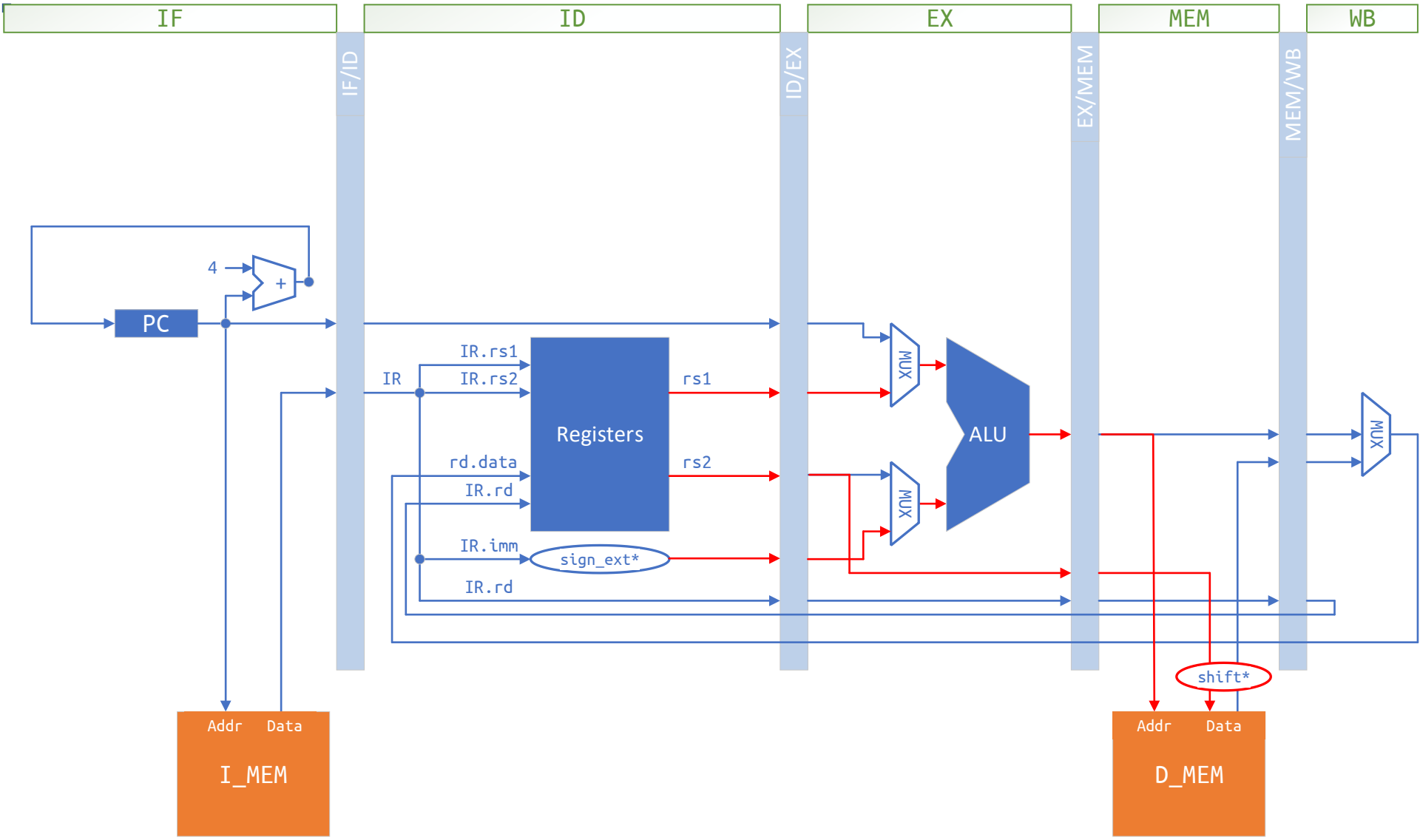


  - rs1: base address register number
  - rs2: source operand register number
  - immediate: offset added to base address
    - Imm split: so that rs1 and rs2 fields are always in the same place

load

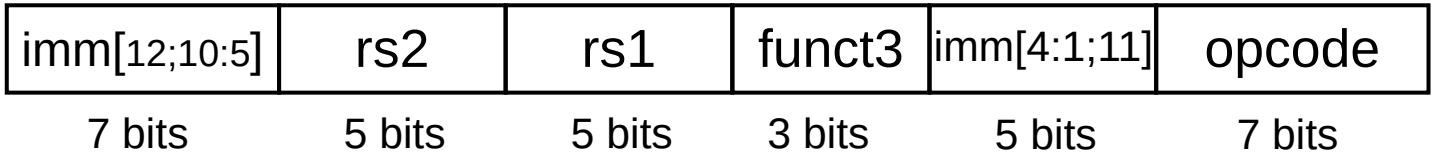

IF
ID
EX
MEM
WB
IF/ID
ID/EX
EX/MEM
MEM/WB
4
+
PC
IR
IR.rs1
IR.rs2
rd.data
IR.rd
IR.imm
IR.rd
Registers
rs1
rs2
sign_ext*
MUX
MUX
ALU
MUX
shift*
Addr
Data
I_MEM
Addr
Data
D_MEM

store

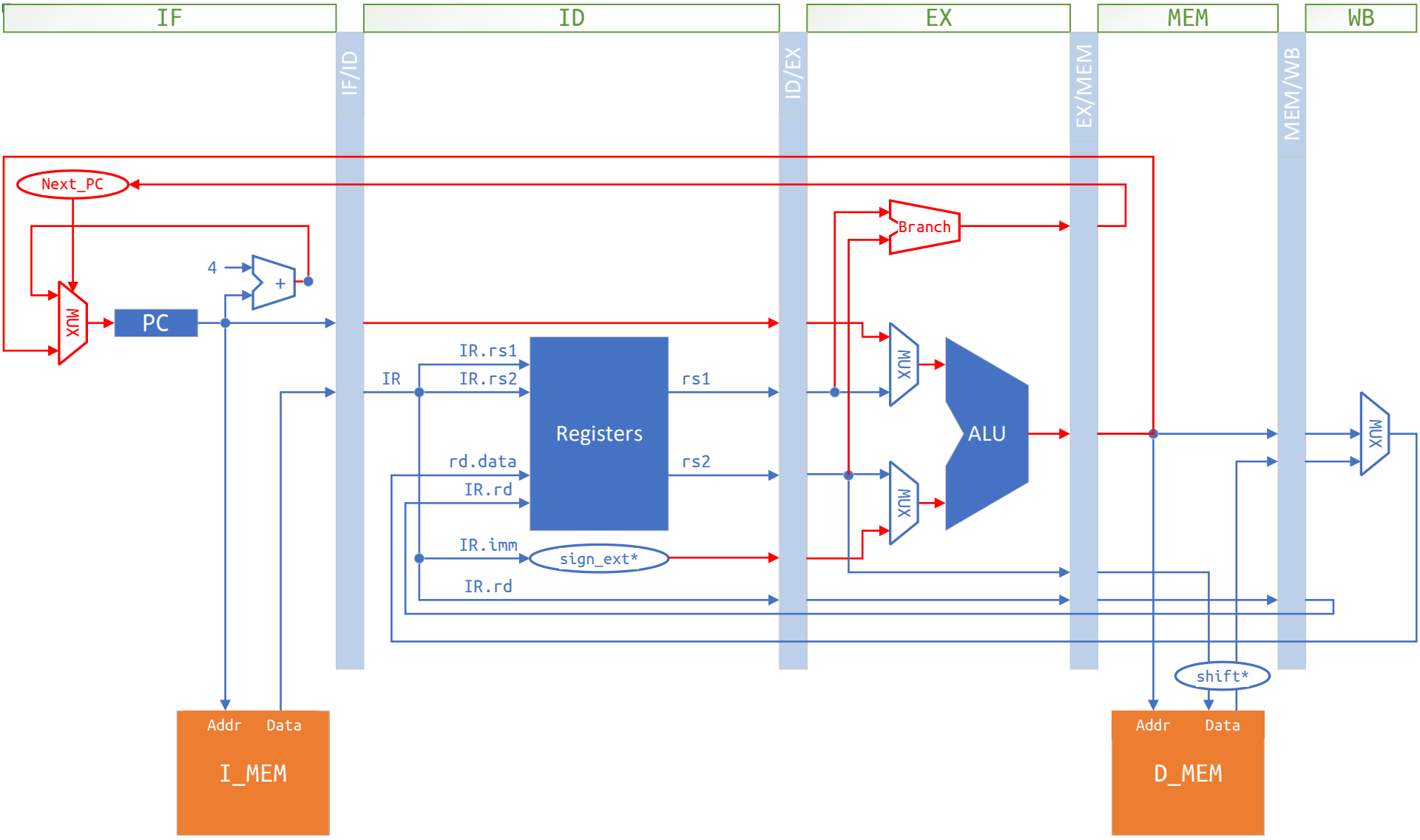

IF
ID
EX
MEM
WB
IF/ID
ID/EX
EX/MEM
MEM/WB
4
+
PC
IR.rs1
IR
IR.rs2
Registers
rs1
rs2
rd.data
IR.rd
IR.imm
sign_ext*
IR.rd
MUX
MUX
ALU
MUX
shift*
Addr
Data
I_MEM
Addr
Data
D_MEM

Put podataka v1.0

- upravljačke naredbe
- Branch
  - B*condition rs1,rs2,label*
  - If (rs1 *condition* rs2) PC= PC + sign_ext(imm13*2)

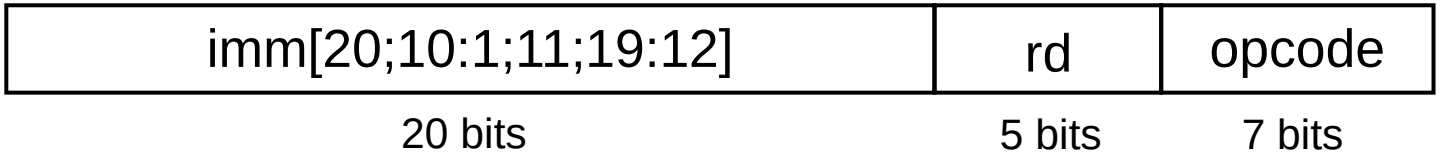


branch

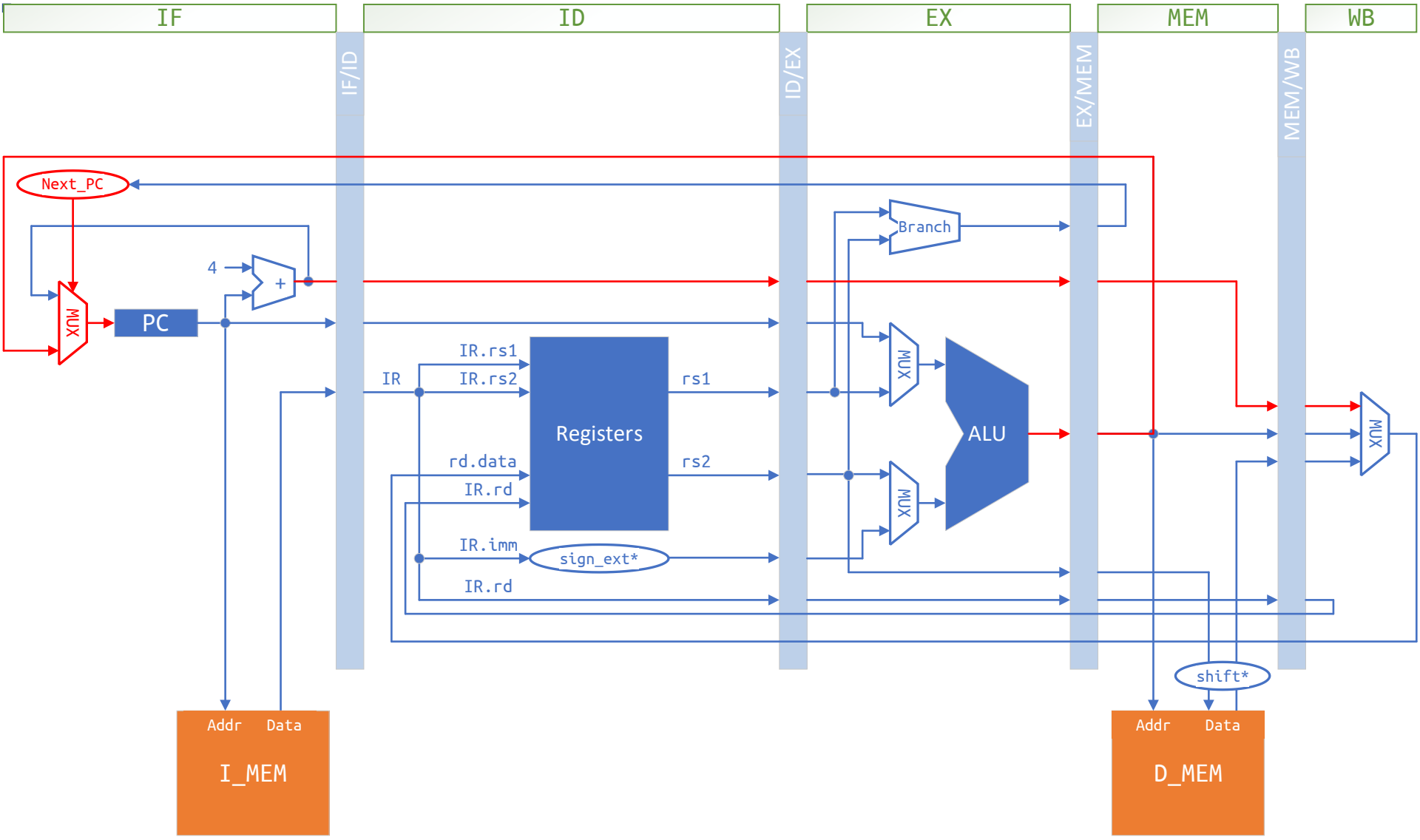

IF
ID
EX
MEM
WB
IF/ID
ID/EX
EX/MEM
MEM/WB
Next_PC
MUX
PC
4
+
Addr
Data
I_MEM
IR
IR.rs1
IR.rs2
rd.data
IR.rd
IR.imm
IR.rd
Registers
rs1
rs2
sign_ext*
Branch
MUX
MUX
MUX
ALU
shift*
Addr
Data
D_MEM
MUX

Put podataka v1.0

- upravljačke naredbe
- jal
  - rd = PC+4; PC= PC + sign_ext(imm20*2)

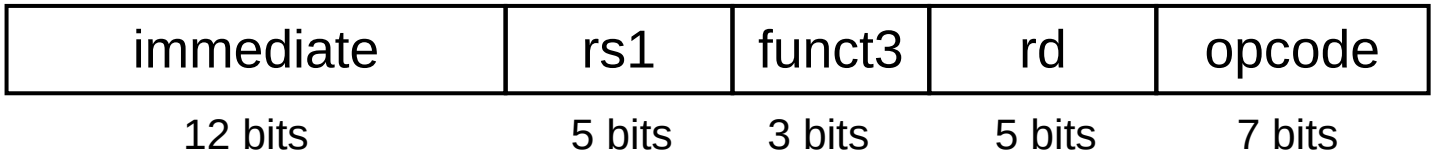


jal

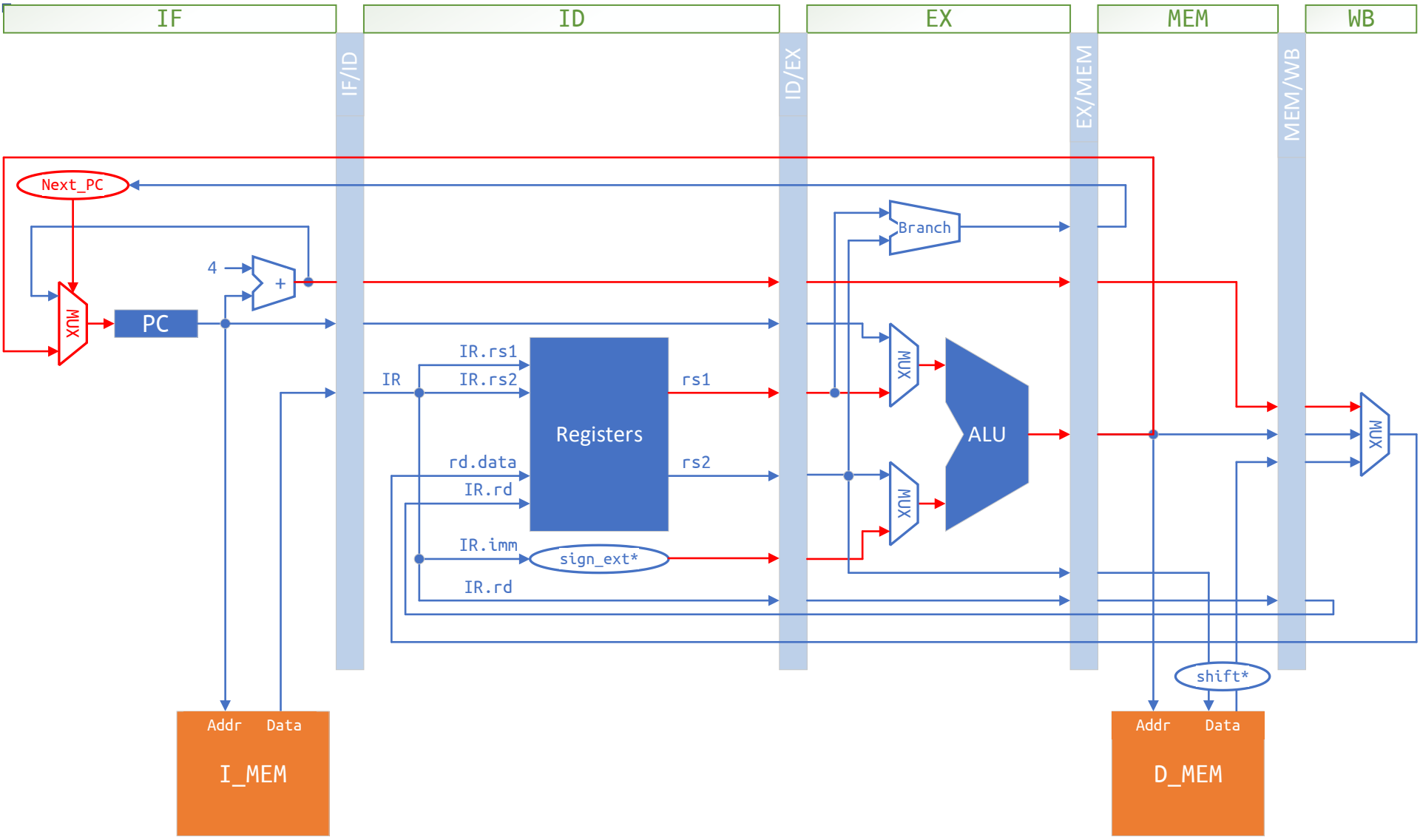

IF
ID
EX
MEM
WB
IF/ID
ID/EX
EX/MEM
MEM/WB
Next_PC
4
+
MUX
PC
Branch
MUX
ALU
MUX
MUX
IR.rs1
IR
IR.rs2
Registers
rs1
rs2
rd.data
IR.rd
IR.imm
sign_ext*
IR.rd
shift*
Addr
Data
I_MEM
Addr
Data
D_MEM

Put podataka v1.0

- upravljačke naredbe
- jalr
  - rd = PC+4; PC =(rs1 + sign_ext(imm12)) & 0xFFFFFFFE



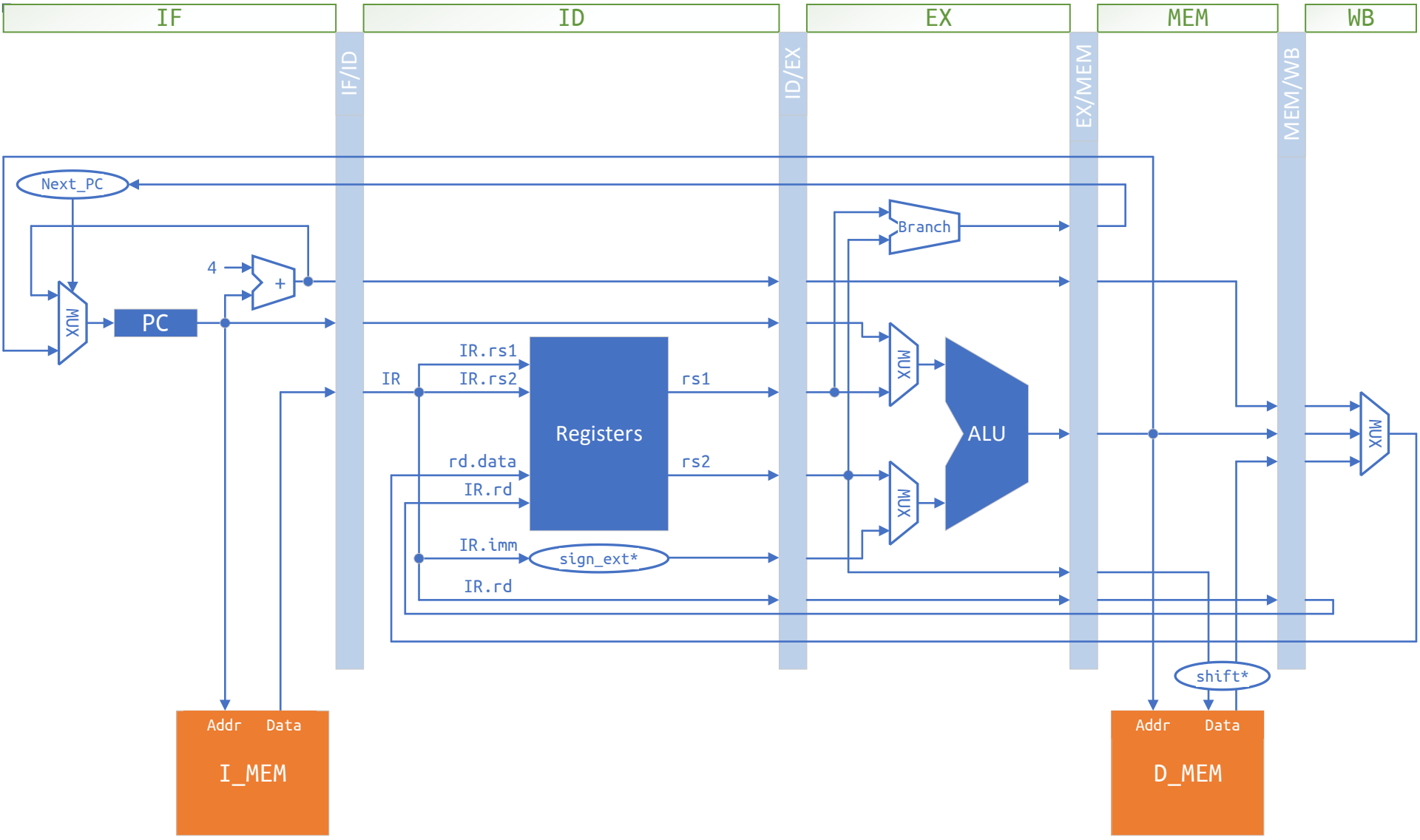
jalr


IF
ID
EX
MEM
WB
IF/ID
ID/EX
EX/MEM
MEM/WB
Next_PC
4
+
MUX
PC
Branch
MUX
ALU
MUX
MUX
IR.rs1
IR
IR.rs2
Registers
rs1
rs2
rd.data
IR.rd
IR.imm
sign_ext*
IR.rd
shift*
Addr
Data
I_MEM
Addr
Data
D_MEM

FRISC-V

V 1.0

Put podataka v1.0



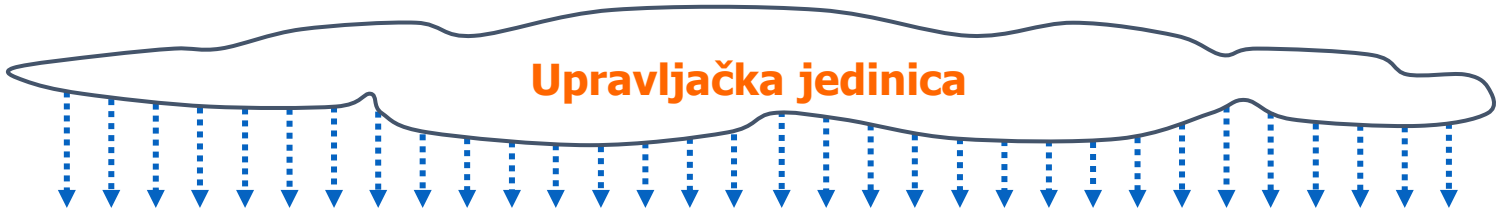
Put podataka v1.0

- Ovime smo definirali cijeli put podataka za naš FRISC-V procesor
  - RV32I arhitektura skupa naredaba
  - Harvard arhitektura pristupa memoriji
  - Protočna arhitektura sa 5 razina
    - IF, ID, EX,MEM,WB

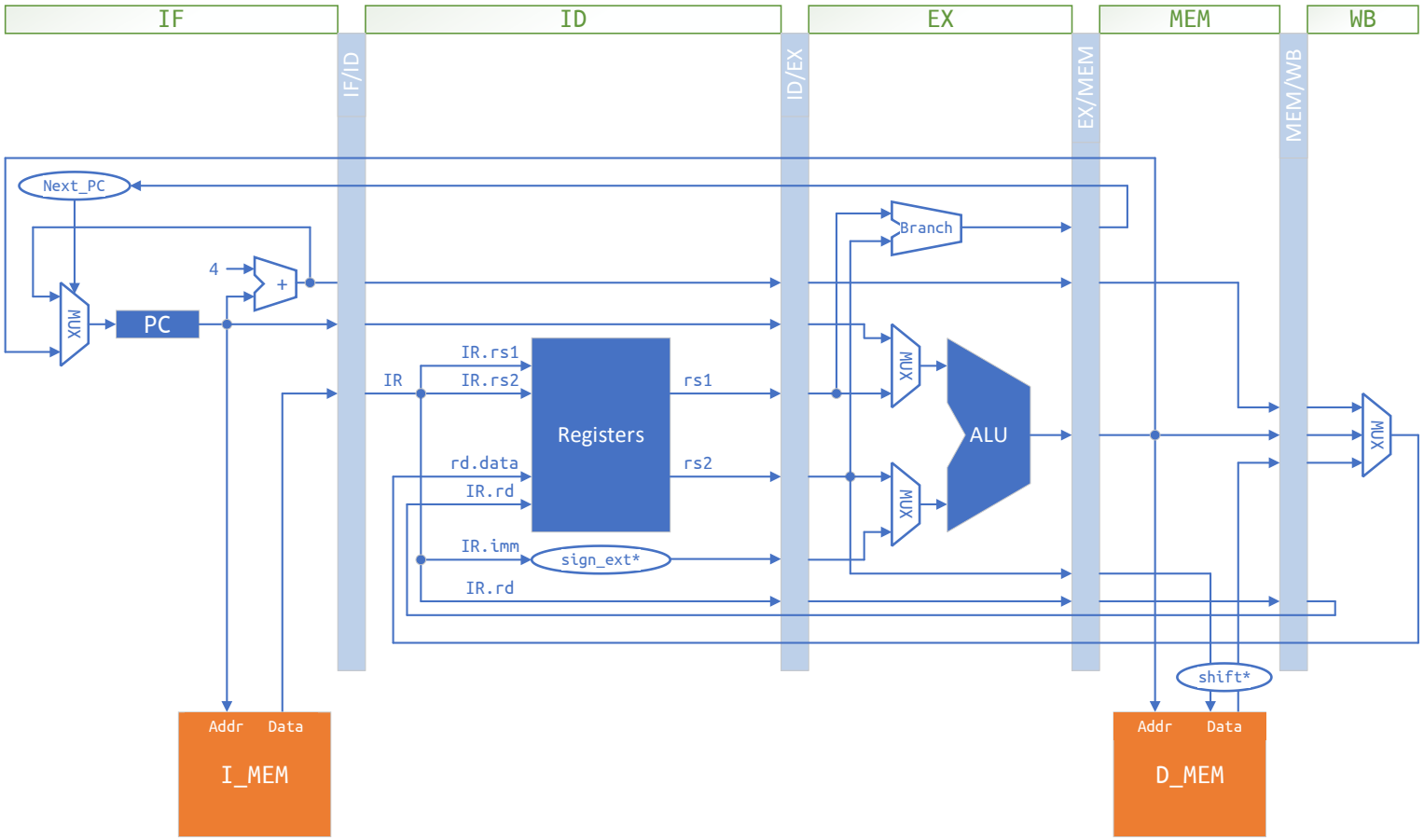
- Upravljačke linije nisu prikazane zbog preglednosti

  - Kao što se može vidjeti iz slike, dijelovi puta podataka prilično su jednostavni,
  - upravljanje s njima također nije kompleksno

Put podataka v1.0



Kako generirati upravljačke signale potrebne za izvođenje opisanih naredaba ?



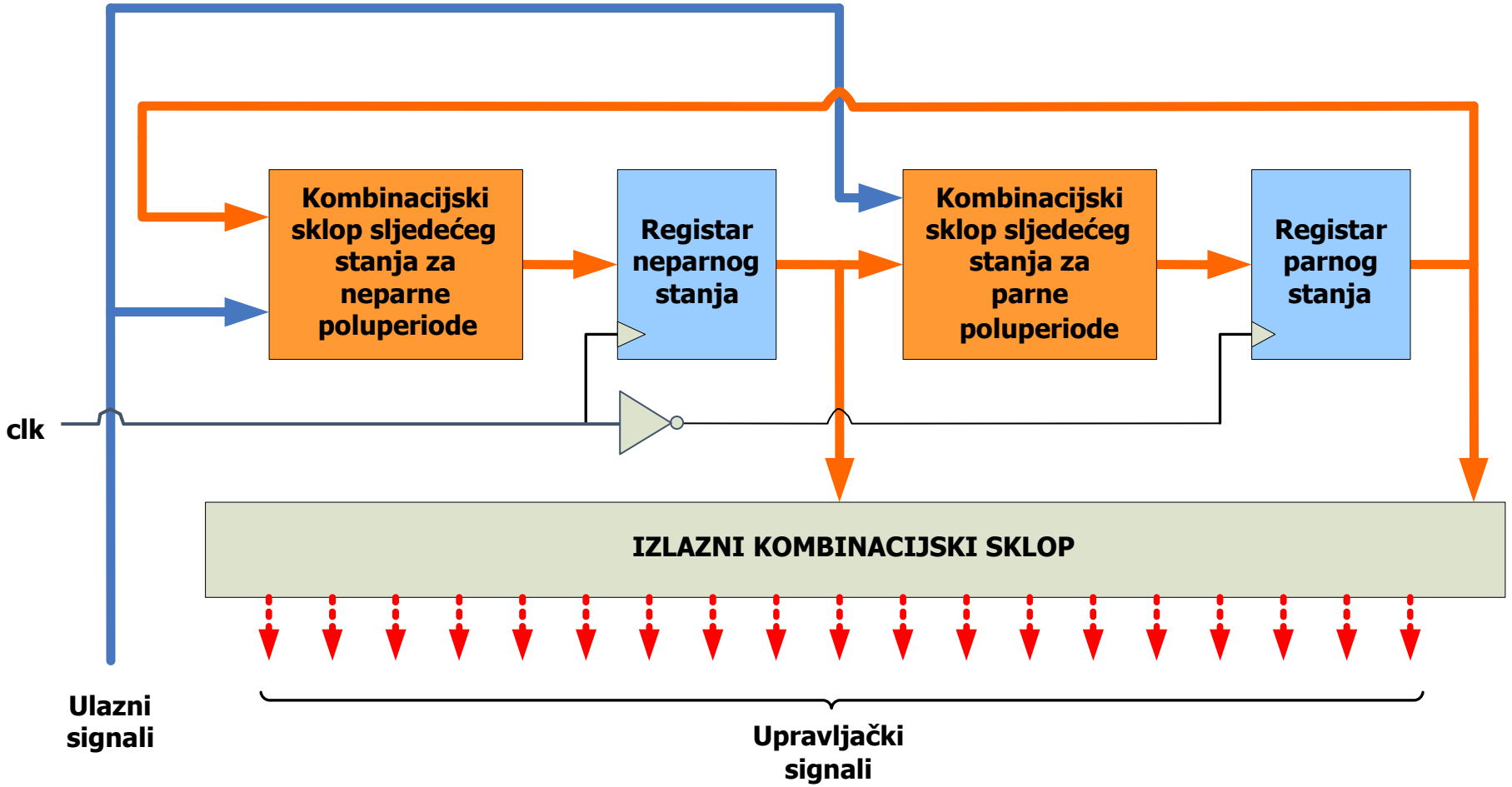
Upravljanje putem podataka

- Kao što ste naučili u "Digitalnoj", jednostavan način generiranja upravljačkih signala može se postići strojem s konačnim brojem stanja (finite state machine - FSM)

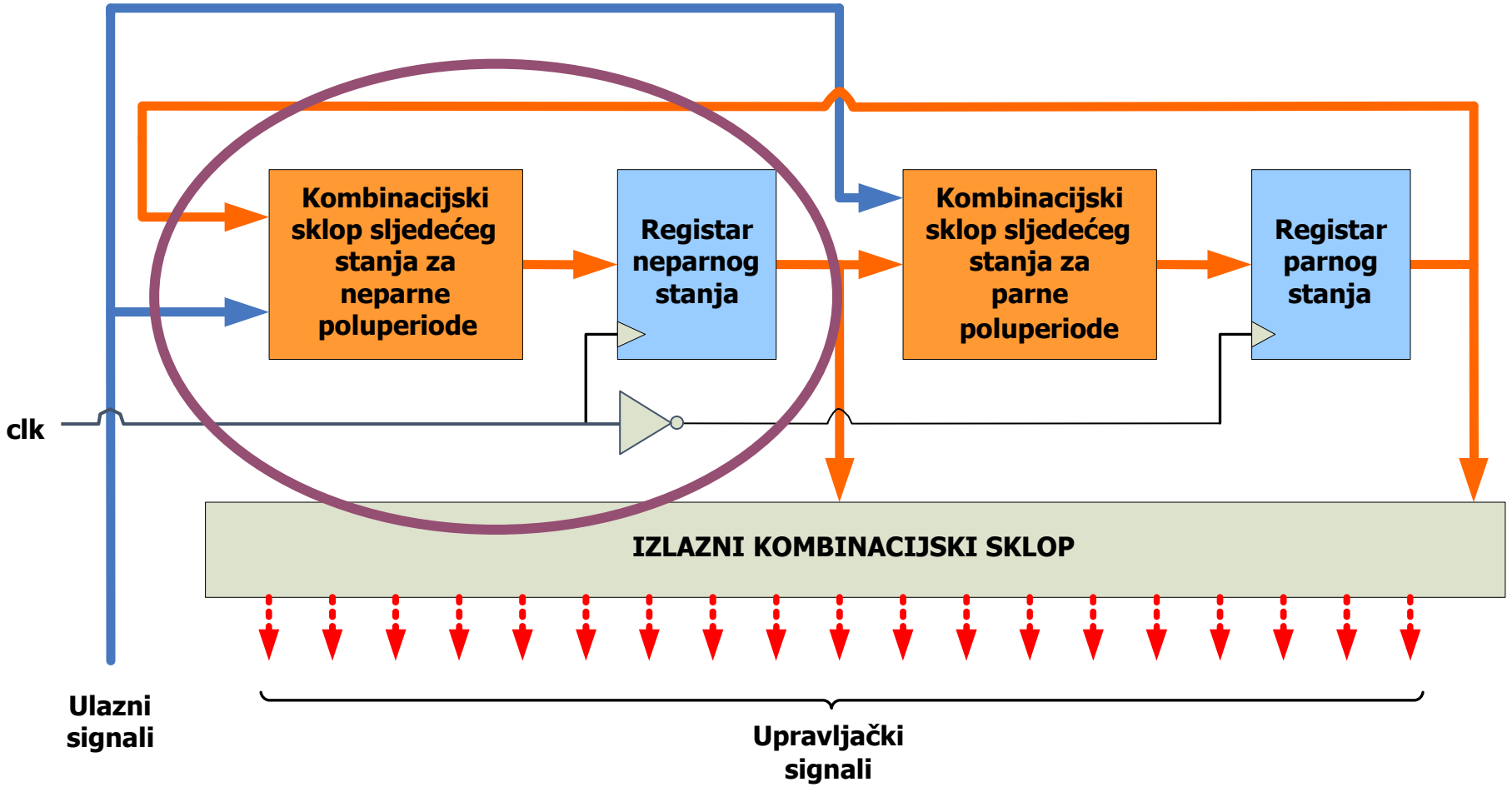
- Pri izvođenju naredaba treba generirati upravljačke signale na rastući i na padajući brid signala vremenskog vođenja clock
- Upravljački signali ovise o:
  - Naredbi koja se izvodi
  - Nekim podacima
  - Prethodnim stanjima puta podataka

- Klasični FSM generira signale na jedan od bridova (rastući ili padajući) pa upravljačka jedinica FRISC-V koristi dvostruki FSM

Upravljačka jedinica – stroj s konačnim brojem stanja

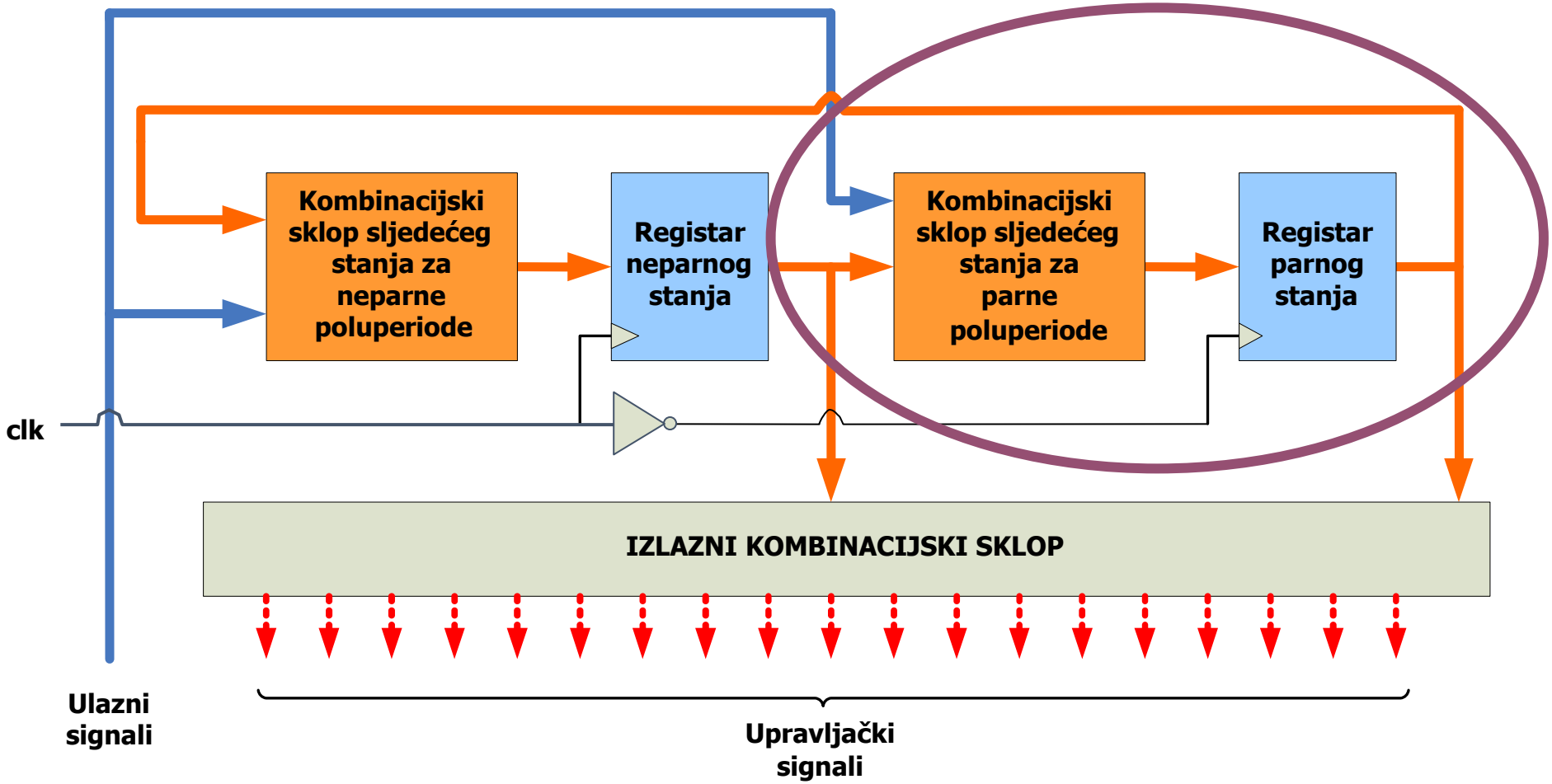


Upravljačka jedinica – stroj s konačnim brojem stanja



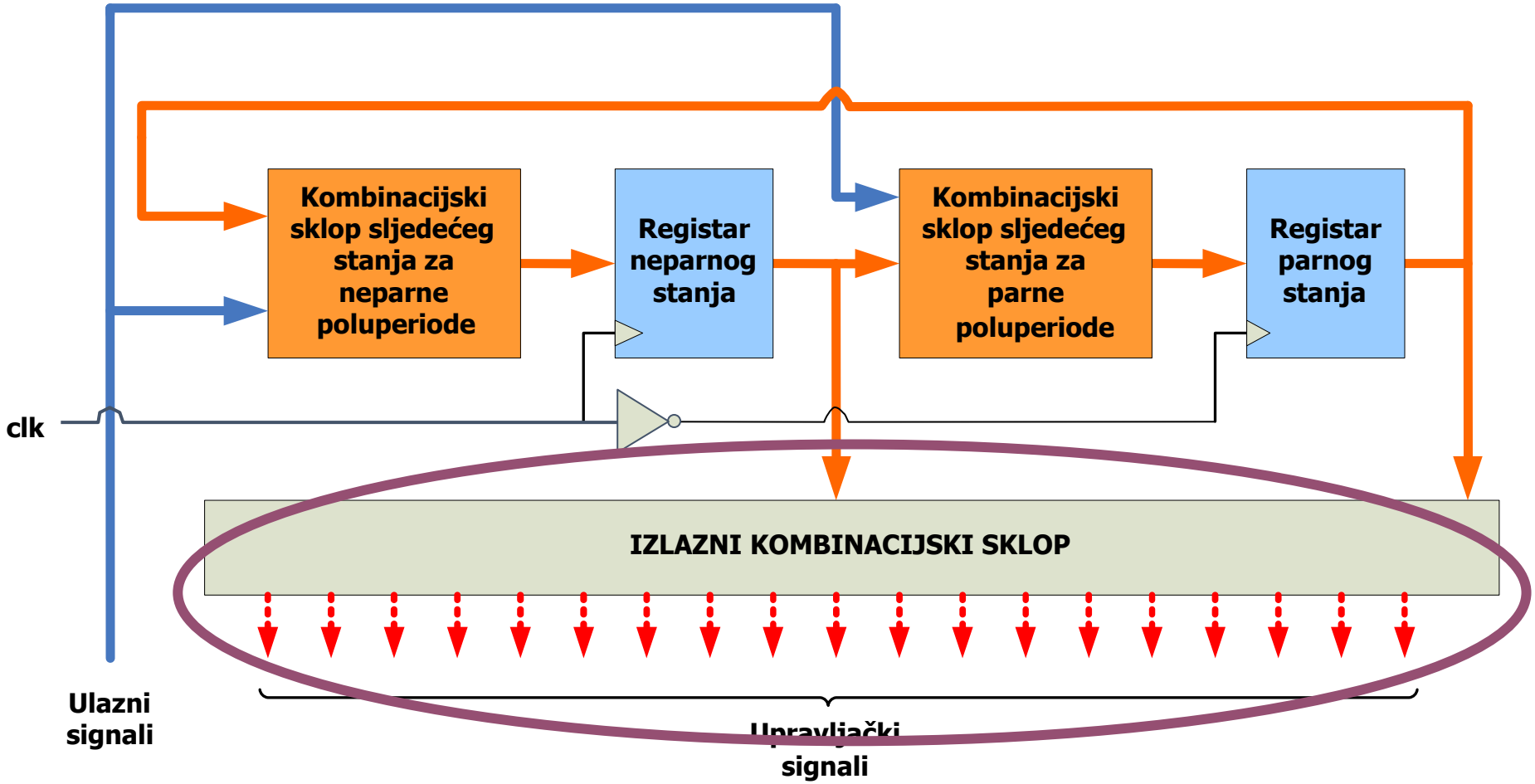
Prvi FSM zadužen je za generiranje svih signala u neparnim poluperiodima

Upravljačka jedinica – stroj s konačnim brojem stanja



Drugi FSM zadužen je za generiranje svih signala u parnim poluperiodima

Upravljačka jedinica – stroj s konačnim brojem stanja



Upravljački signali generiraju se na temelju stanja oba stroja