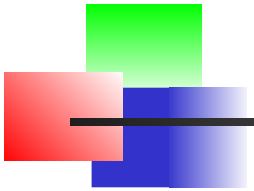
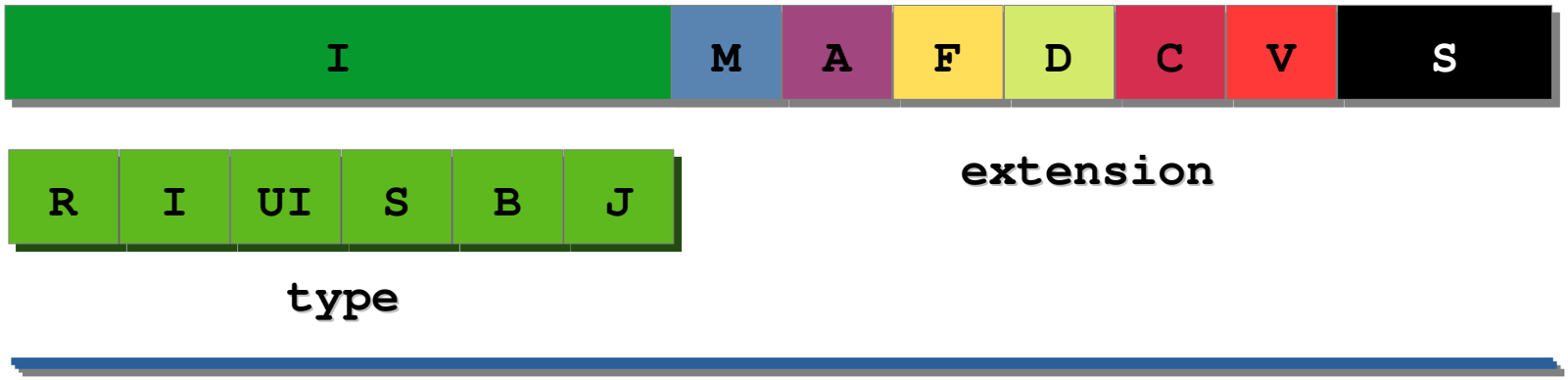


“RISC-V design”

“Rise of RISC-V: The computer chip design you need to know about”

RISC-V : Hardware Design



RTL – Verilog : busses,registers,ALUs,memories,decoders,..

Logic – Verilog: and,or,xor,nand,..

FPGA

masks/ASIC

design

RISC-V : Software-Hardware

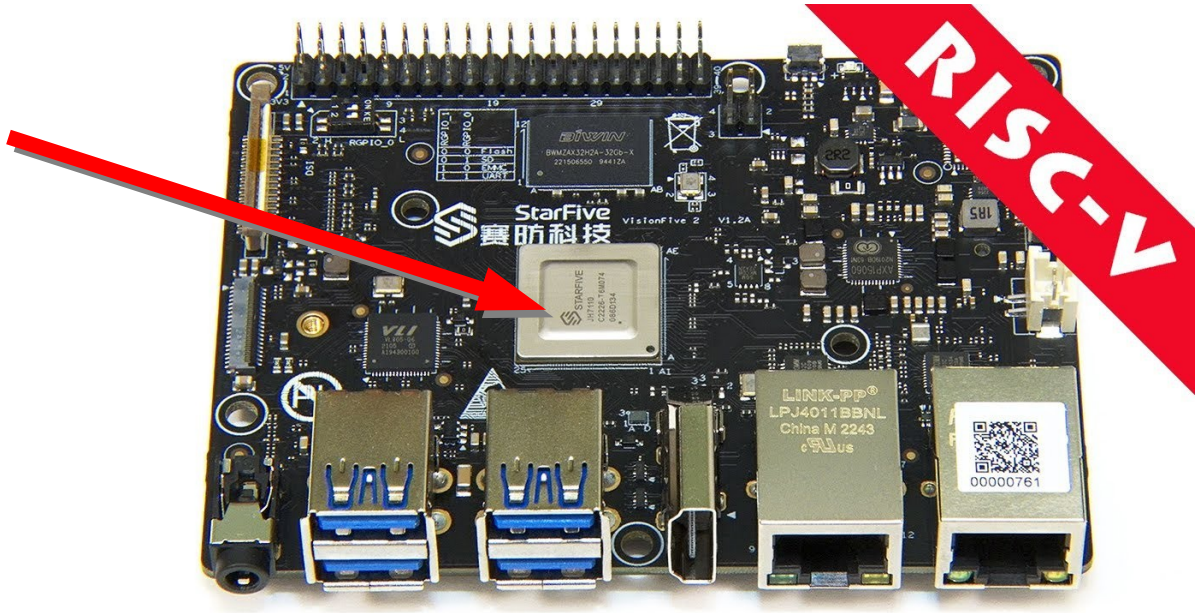
```
module ALU(Out,A,B,S,cin);
    input [3:0] A, B;
    input [2:0] S;
    input cin;
    output [3:0] Out;
    reg [3:0] Out;

    always @(S or A or B or cin)
        case (S)
        0 : Out = 4'b0000;
        1 : Out = B - A - cin;
        2 : Out = A - B - cin;
        3 : Out = A + B + cin;
        4 : Out = A ^ B;
        5 : Out = A | B;
        6 : Out = A & B;
        7 : Out = 4'b1111;
        endcase
endmodule
```

RTL – design

StarFive VisionFive 2 design

VisionFive 2 is high-performance RISC-V single board computer (SBC) with an integrated GPU.

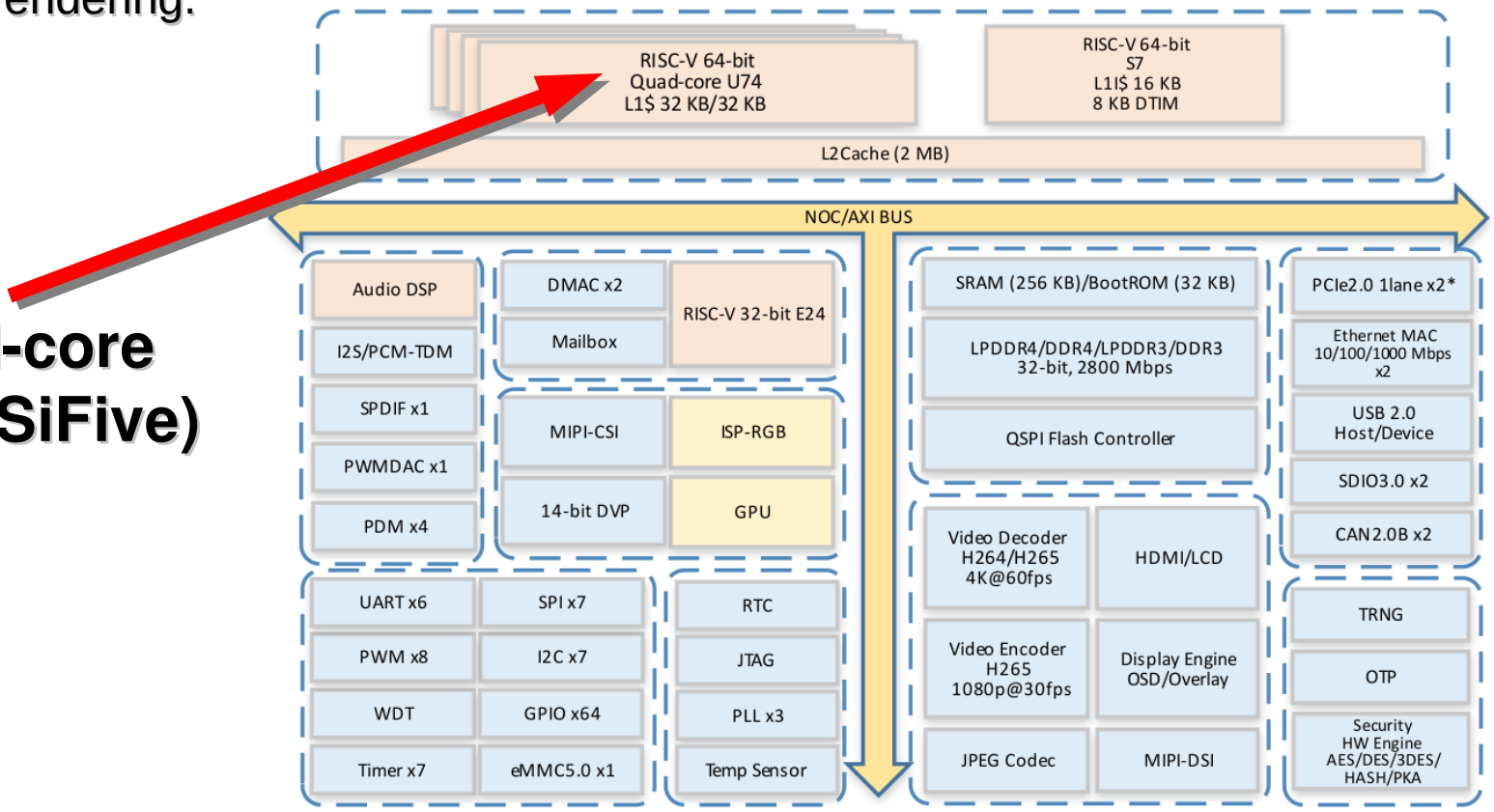
JH7110 (SiFive)



StarFive - JH7110

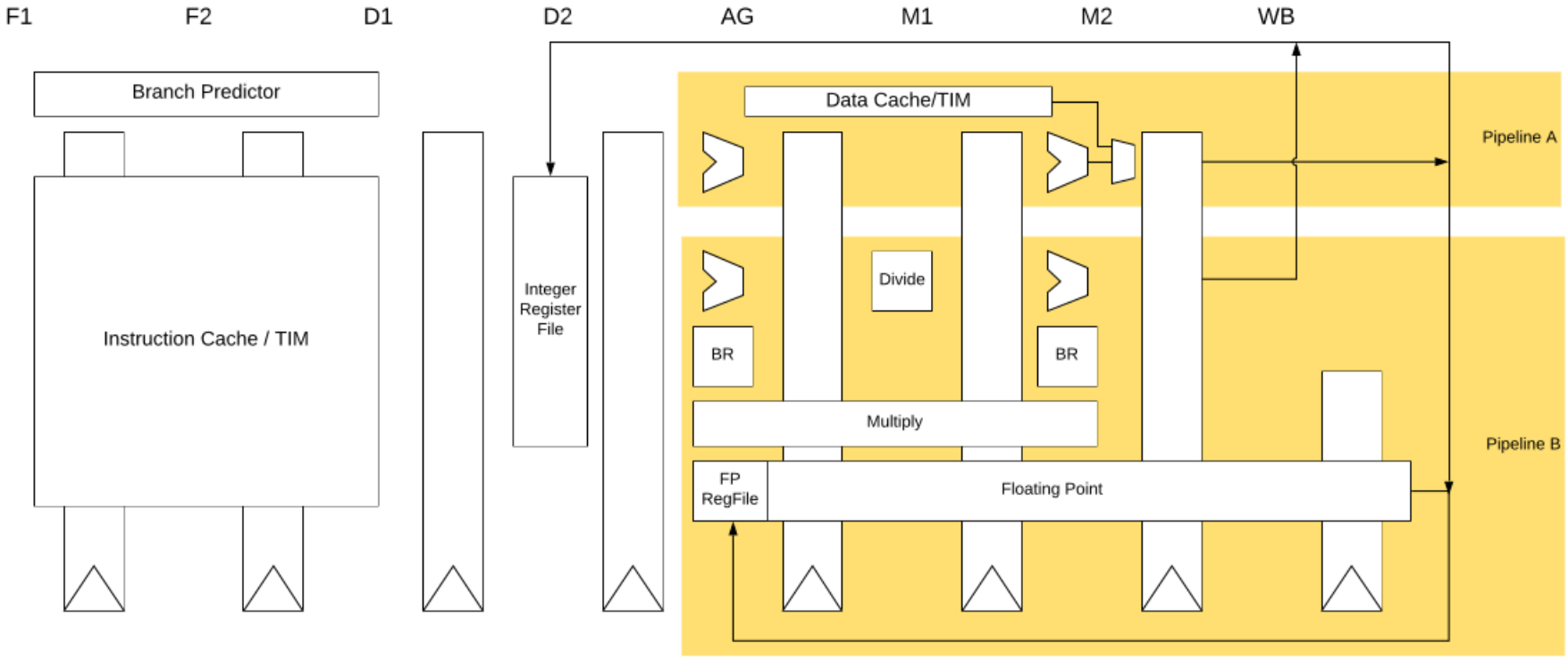
JH7110 is a high-performance **RISC-V SoC with 64 bit quad-core** processor (**SiFive U74**) and 1.5GHz frequency. It integrates high-speed interfaces and GPU, enabling stronger image processing capabilities, such as 3D rendering.

Quad-core U74 (SiFive)

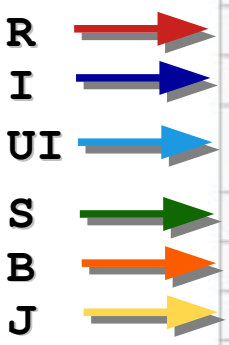


SiFive U74 pipeline – 8 stages

SiFive U74 - The S7 execution unit is a dual-issue, in-order pipeline (8-stages).
- instruction fetch (**F1** and **F2**),
- instruction decode (**D1** and **D2**), address generation (**AG**),
- data memory access (**M1** and **M2**), and register write-back (**WB**).



RISC-V ISA - formats



32-bit RISC-V Instruction Formats

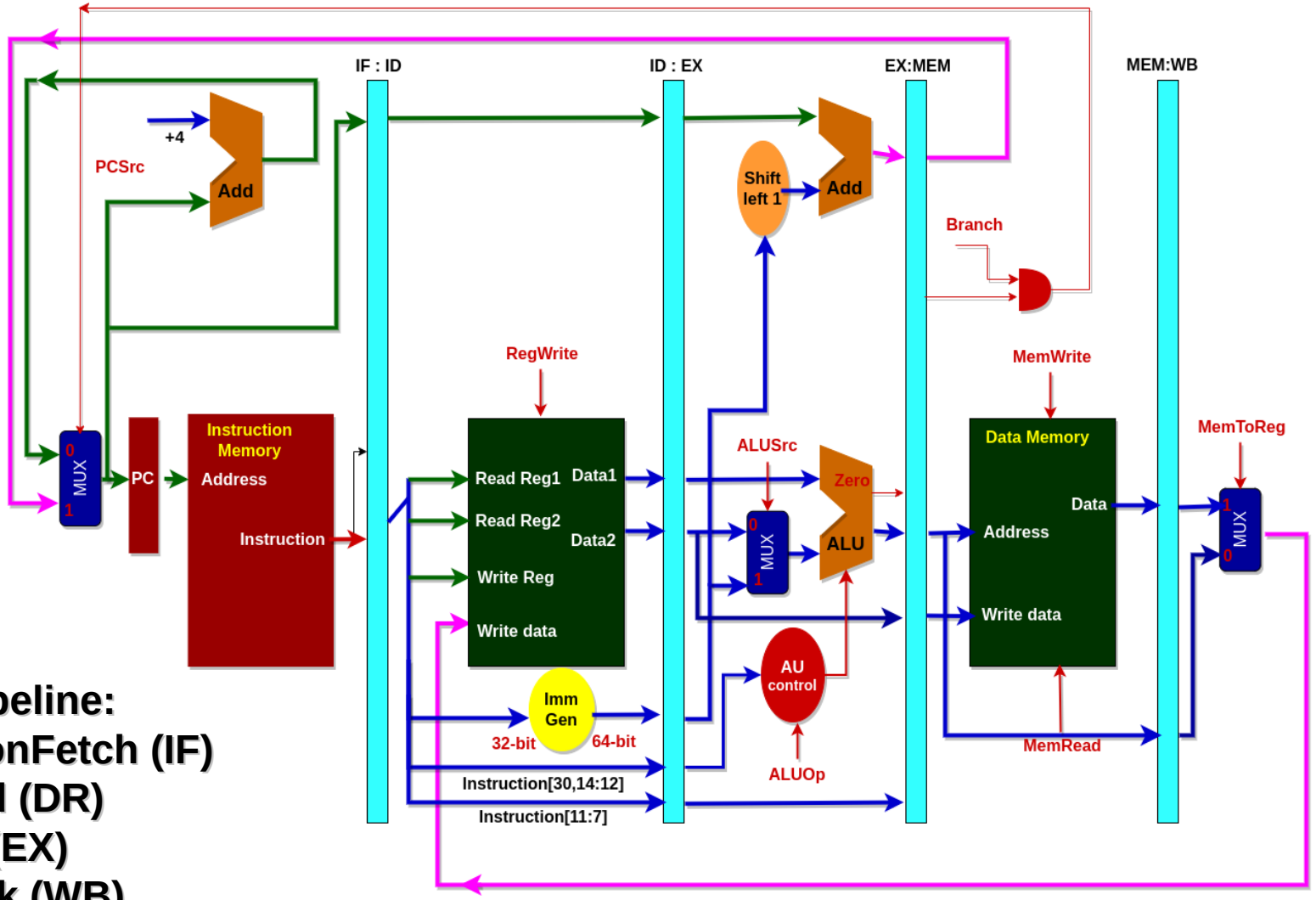
	Instruction Formats	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Register/register	funct7							rs2					rs1					funct3			rd						opcode						
I	Immediate	imm[11:0]												rs1					funct3			rd						opcode						
UI	Upper Immediate	imm[31:12]																				rd						opcode						
S	Store	imm[11:5]							rs2					rs1					funct3			imm[4:0]						opcode						
B	Branch	[12]	imm[10:5]						rs2					rs1					funct3			imm[4:1]				[11]		opcode						
J	Jump	[20]	imm[10:1]										[11]	imm[19:12]								rd						opcode						

- **opcode (7 bit):** partially specifies which of the 6 types of *instruction formats*
- **funct7 + funct3 (10 bit):** combined with **opcode**, these two fields describe what operation to perform
- **rs1 (5 bit):** specifies register containing first operand
- **rs2 (5 bit):** specifies second register operand
- **rd (5 bit):**: Destination register specifies register which will receive result of computation

ISA *computational instructions* use a three-operand format, in which the first operand is the destination register, the second operand is a source register, and the third operand is either a second source register or an immediate value. This is an example three-operand instruction:

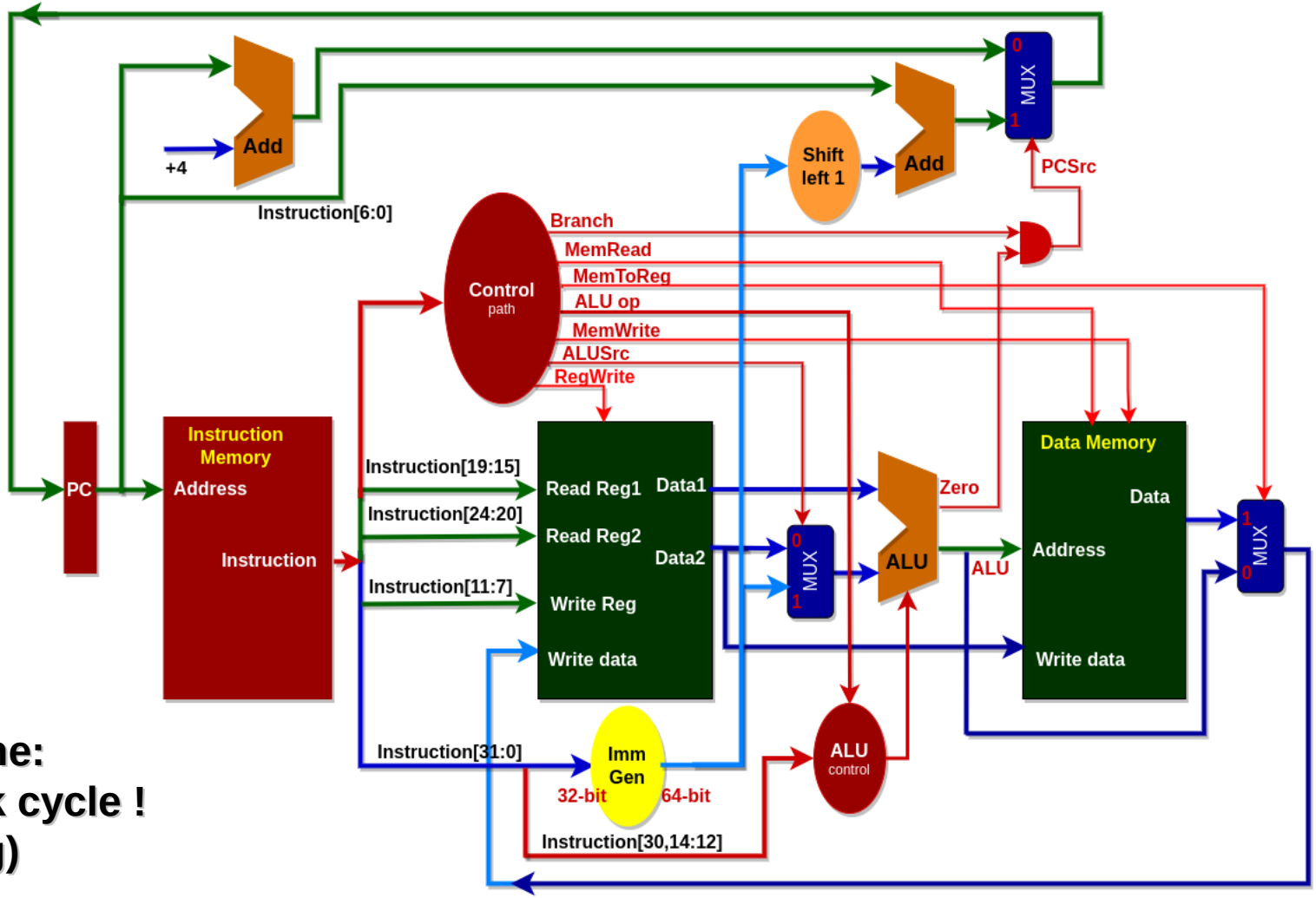
```
add x1,x2,x3
```

Our design : with pipeline



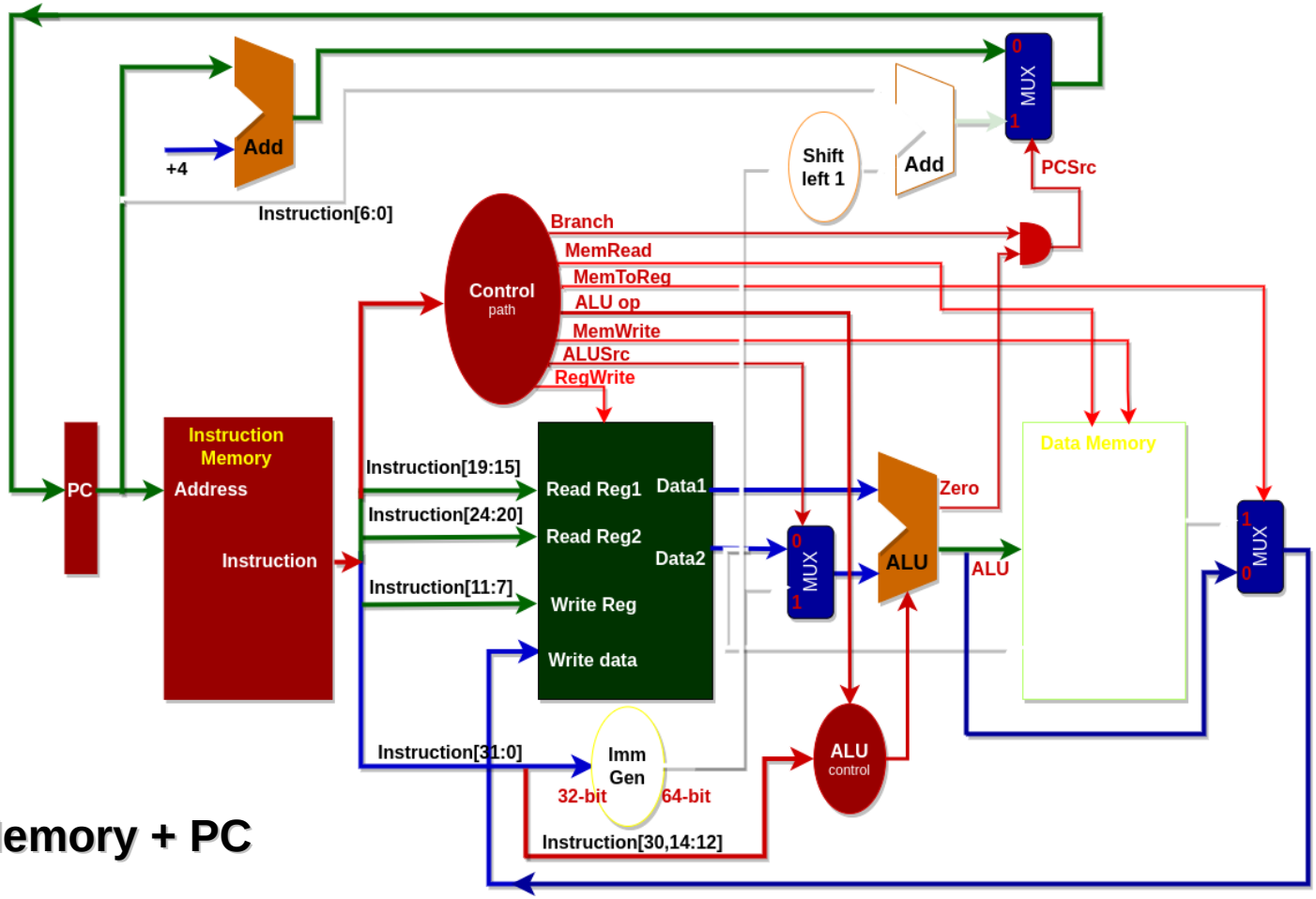
4-stage pipeline:
- **InstructionFetch (IF)**
- **DataRead (DR)**
- **Execute (EX)**
- **WriteBack (WB)**

Our design – without pipeline



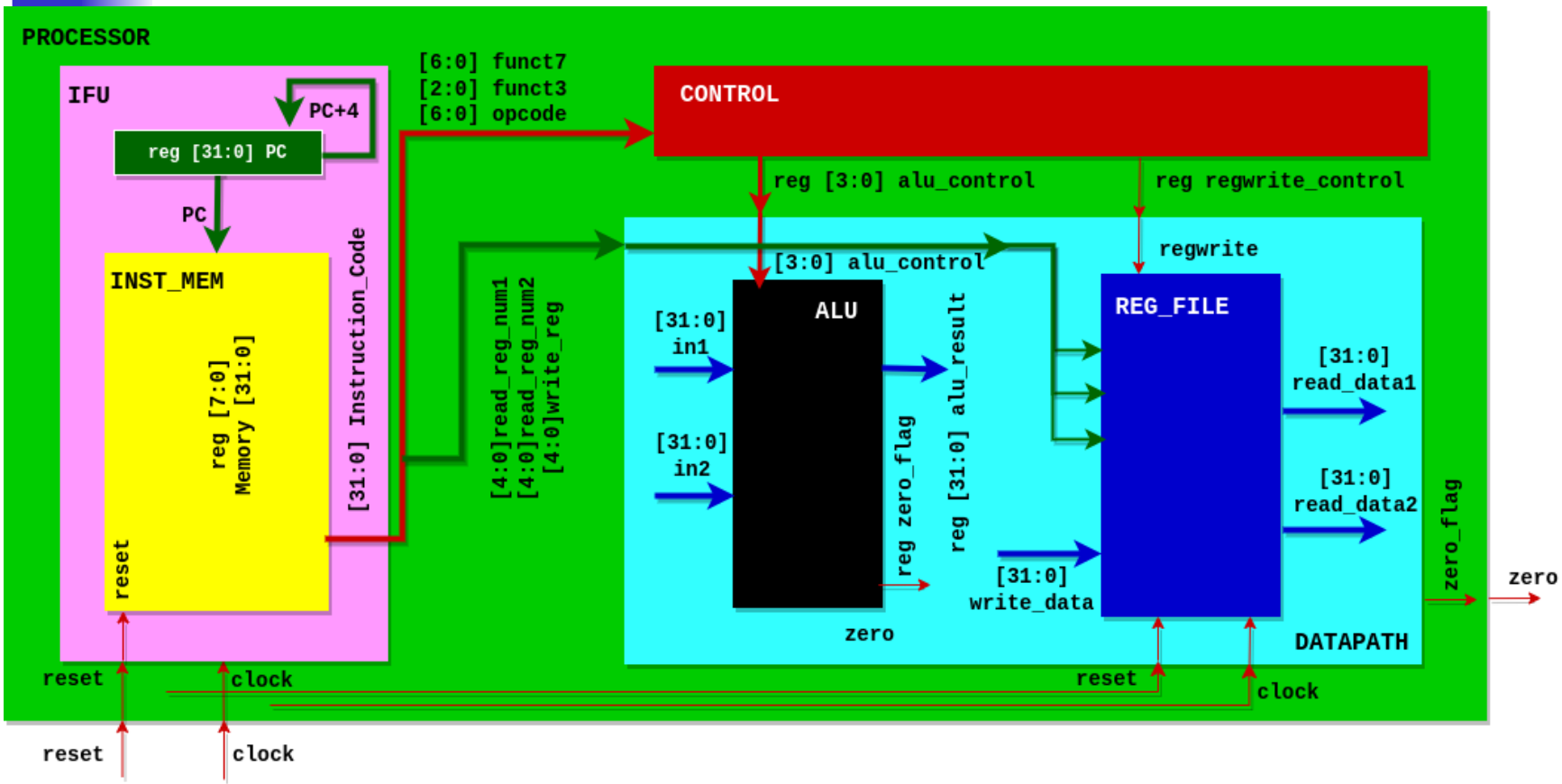
No pipeline:
One clock cycle !
(very long)

Our design: R-type only



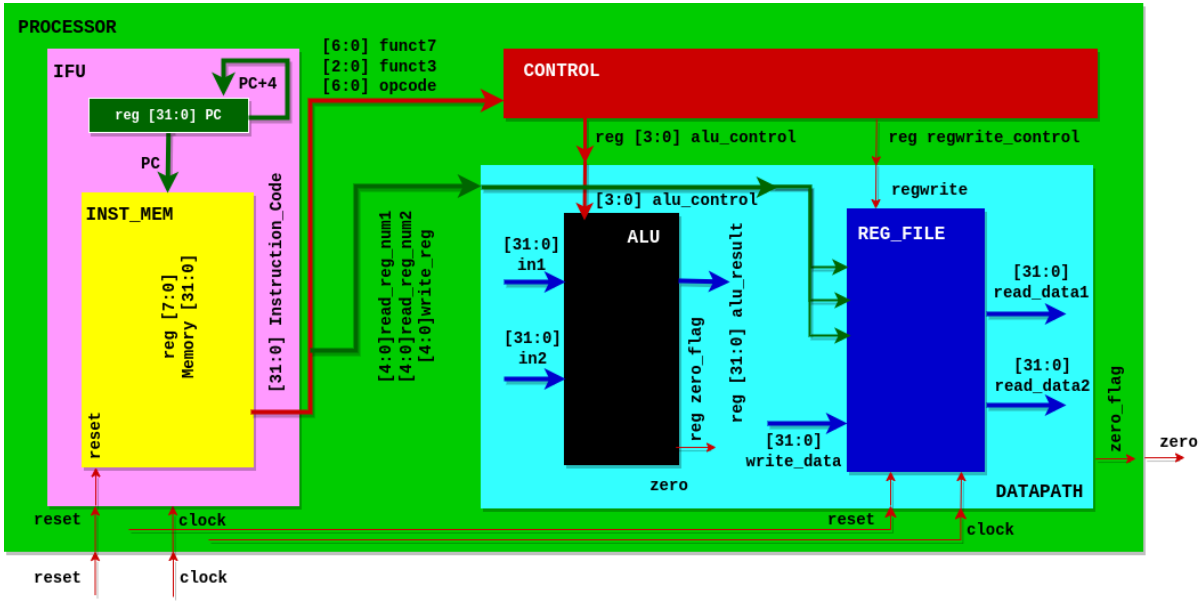
R-type only:
Instruction Memory + PC
Registers
ALU

Our design : Verilog model



IFU : Instruction Fetch Unit (INST_MEM: Instruction Memory + PC)
DATAPATH: ALU + REG_FILE: Register File
CONTROL

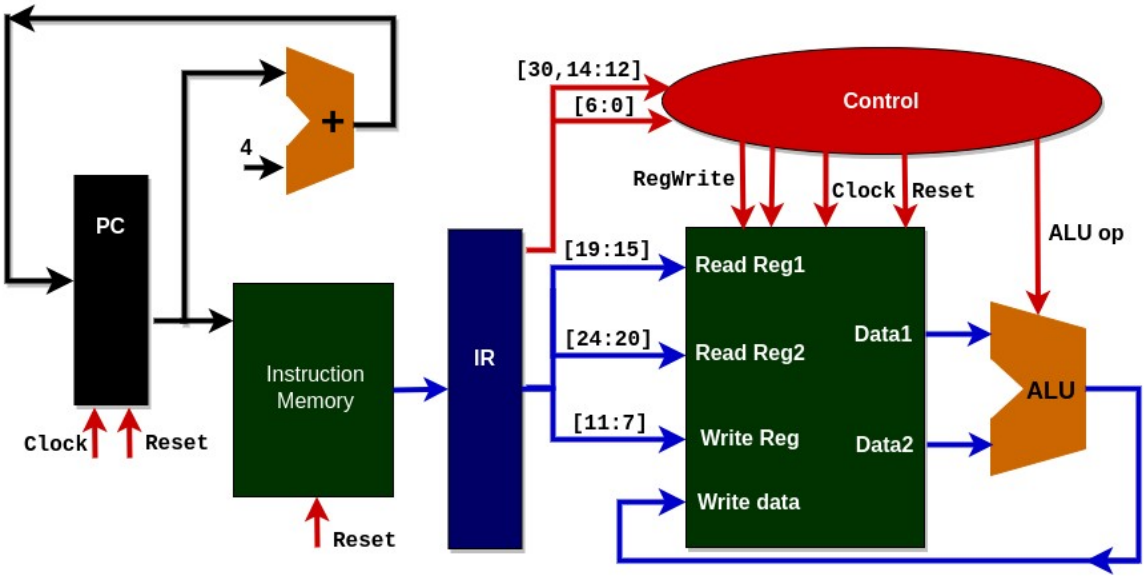
Our design: PROCESSOR




```
`include "CONTROL.v"
`include "DATAPATH.v"
`include "IFU.v"

module PROCESSOR(
    input clock,
    input reset,
    output zero
);

wire [31:0] instruction_code;
wire [3:0] alu_control;
wire regwrite;

IFU IFU_module(clock, reset, instruction_code);

CONTROL control_module(instruction_code[31:25],instruction_code[14:12],
                       instruction_code[6:0],alu_control, regwrite);

DATAPATH datapath_module(instruction_code[19:15], instruction_code[24:20],
                  instruction_code[11:7], alu_control, regwrite, clock, reset, zero);

endmodule
```

Our design: Lab3

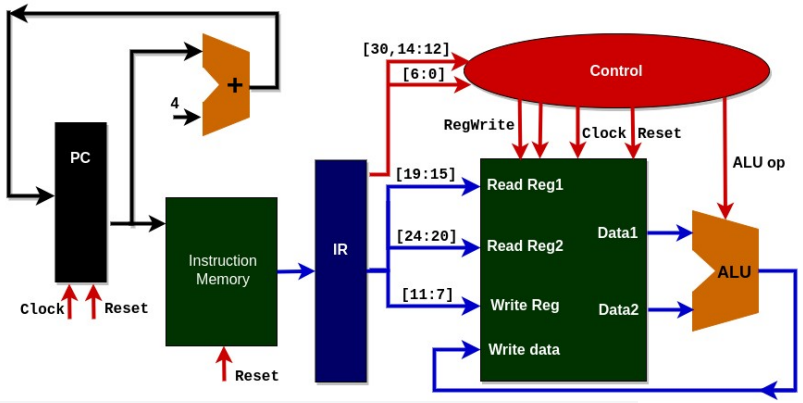
Lab3
with
iverilog
and
gtkwave

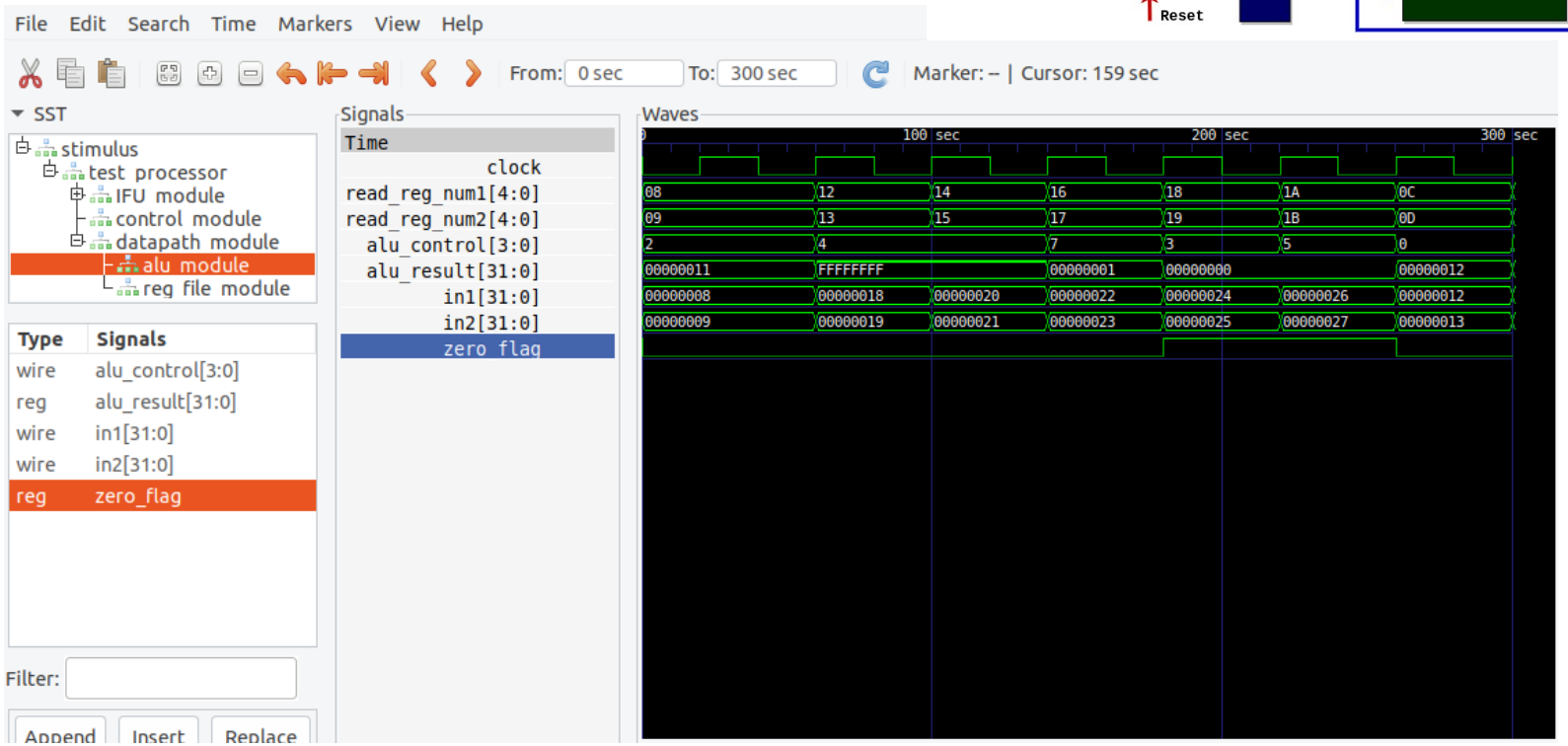



```
bako@lpi4a:~/lab3$iverilog riscv.v riscv_tb.v -o riscv_model

bako@lpi4a:~/lab3$vvp riscv_model

bako@lpi4a:~/lab3$gtkwave riscv_wave.vcd
```

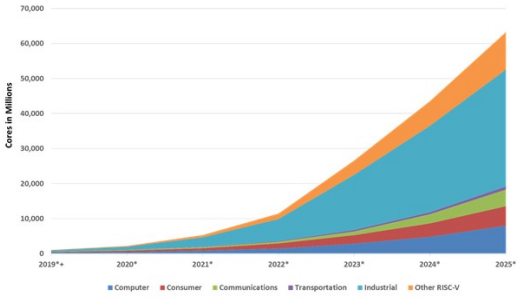
Our design: Lab3

GTKWave waveforms





Evaluation !



Write min 2 pages on topic:

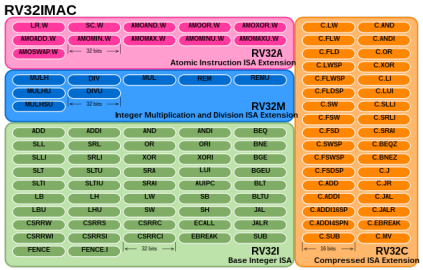
RISC-V est notre avenir !

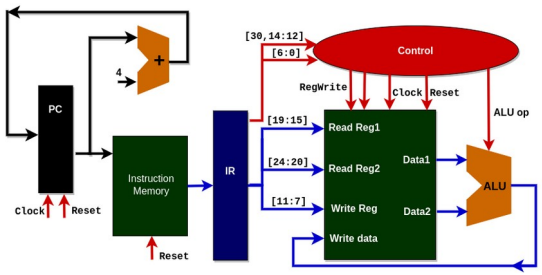
RISC-V is our future !

RISC-V是我们的未来!

RISC-V shì wǒmen de wèilái!

ISA	Chips?	Architecture License?	Commercial Core IP?	Add Own Instructions?	Open-Source Core IP?
x86	Yes, *three* vendors	*No*	*No*	*No*	*No*
ARM	Yes, *many* vendors	Yes, *expensive*	Yes, *one* vendor	*No (Mostly)*	*No*
RISC-V	Yes, *many* vendors	Yes, *free*	Yes, *many* vendors	*Yes*	*Yes, many available*





SmartComputerLab