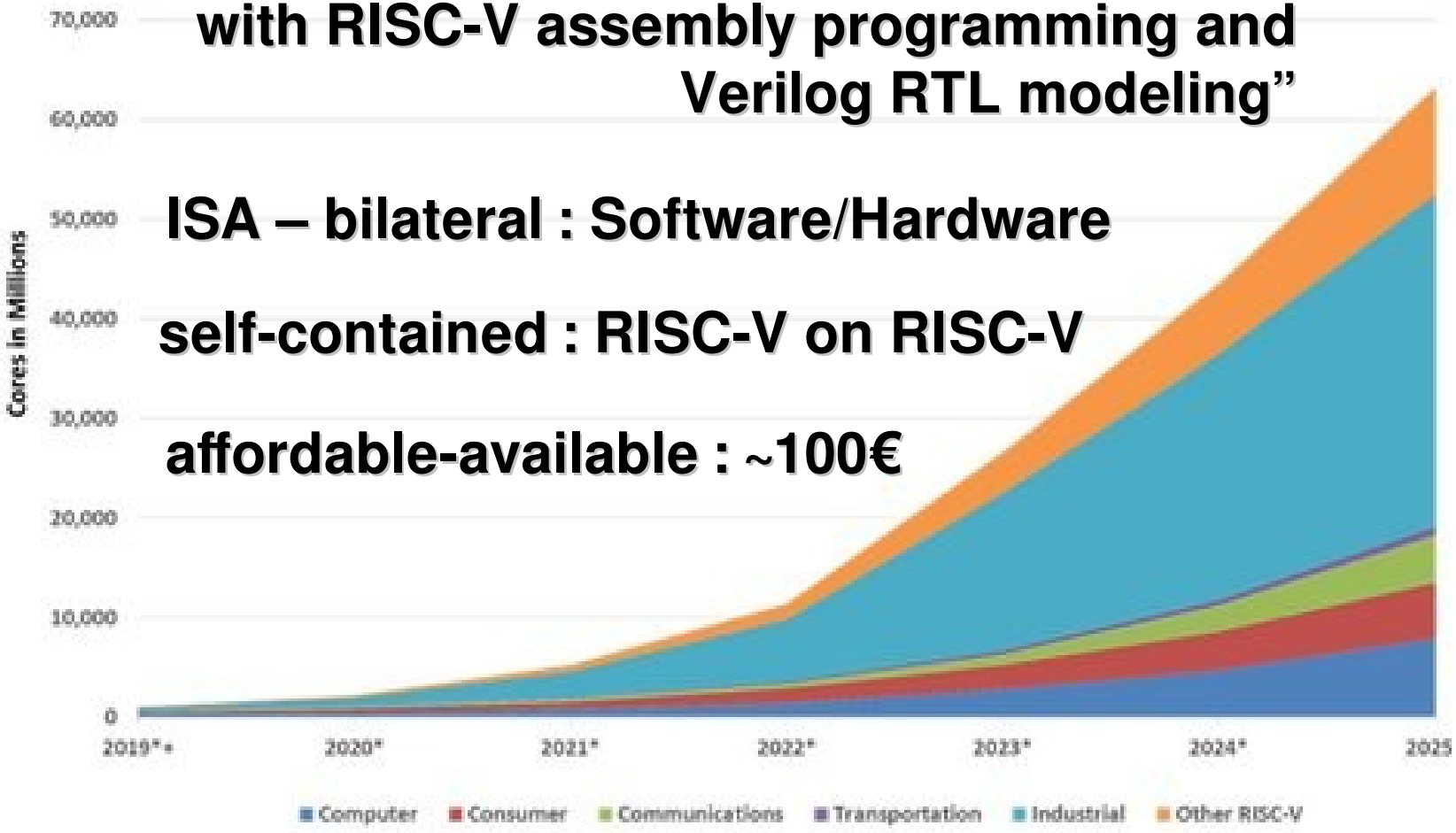
“Teaching RISC-V ISA

with RISC-V assembly programming and Verilog RTL modeling”

ISA – bilateral : Software/Hardware

self-contained : RISC-V on RISC-V

affordable-available : ~100€



6.8mln/hour

Source: Semico Research Corp.

RISC-V : Software Programming

C programming + intrinsics (V)

Assembly level programming

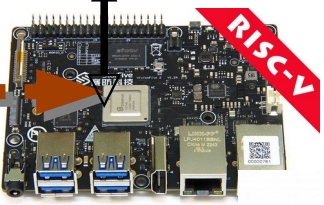
USER – binaries

RV64

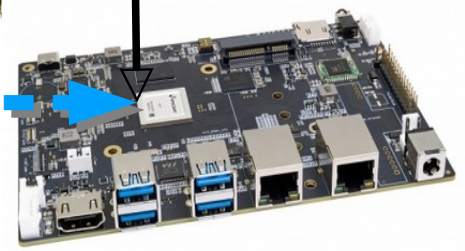
OS

I	M	A	F	D	C	V	B	AI	S

StarFive JH7110
SiFive U74

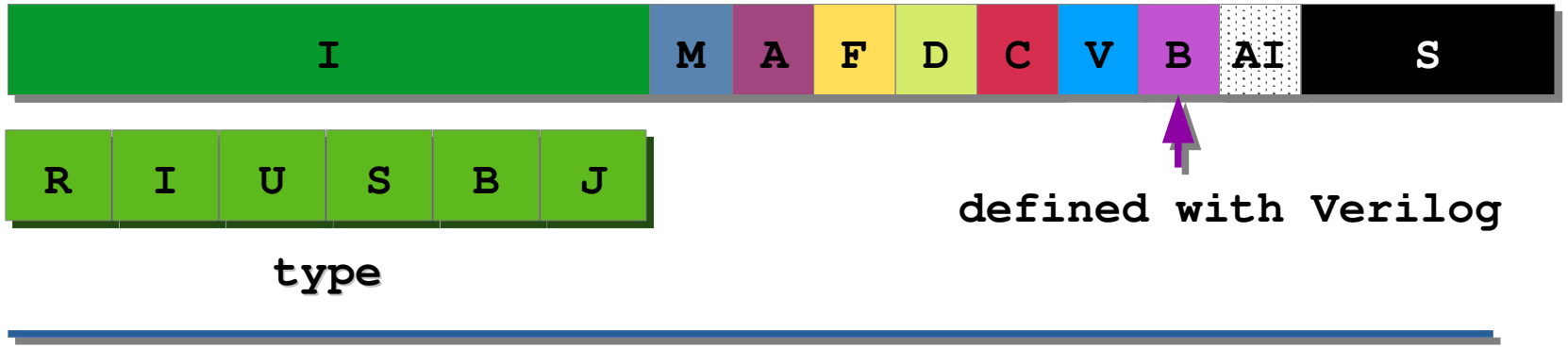


SpacemiT K1 – X60



PLabs

RISC-V : Hardware Design/Modeling

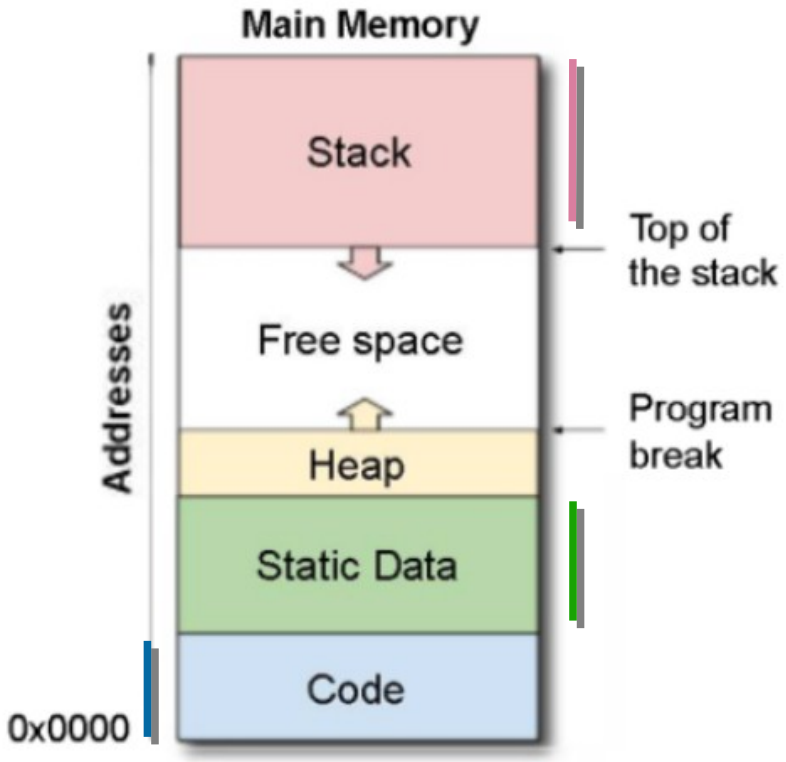


RTL – Verilog : busses,registers,ALUs,memories,decoders,..

Logic – Verilog: and,or,xor,nand,..

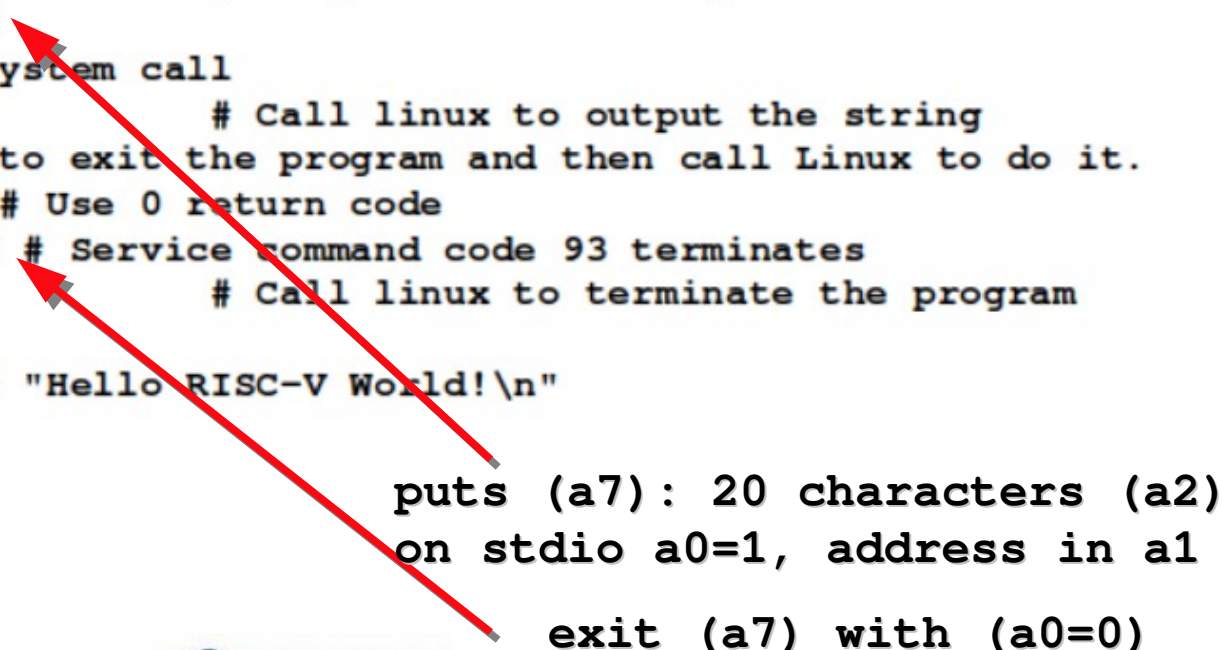
PLabs : Programming Labs

```
.LC0:
        .string "HelloWorld"
        .text
        .align  1
        .globl  main
        .type   main, @function
main:
.LFB6:
        addi    sp,sp,-32
        sd      ra,24(sp)
        sd      s0,16(sp)
        addi    s0,sp,32
        mv      a5,a0
        sd      a1,-32(s0)
        sw      a5,-20(s0)
        lla     a0,.LC0
        call    puts@plt
        li      a0,0
        call    exit@plt
```



PLabs : system calls

```
.global main                        # Provide program starting address to linker
# Setup the parameters to print hello world and then call Linux to do it.
main:
        addi a0, x0, 1              # 1 =  stdout
        la a1, helloworld           # load address of helloworld
        addi a2, x0, 20             # length of our string
        addi a7, x0, 64
        # linux write system call
        ecall                       # Call linux to output the string
# Setup the parameters to exit the program and then call Linux to do it.
        addi a0, x0, 0 # Use 0 return code
        addi a7, x0, 93 # Service command code 93 terminates
        ecall                       # Call linux to terminate the program
.data
helloworld:      .ascii "Hello RISC-V World!\n"
```



PLabs : arithmetics

MultTen.s

```
.data
        .string "Result=%d\n"
.globl  main

main:
        li      a5,4
        mv      a4,a5
        slliw   a5,a5,2
        addw    a5,a5,a4
        slliw   a5,a5,1
        mv      a1,a5
        lla     a0,.data
        call    printf@plt  # link to physical address
        li      a0,0
        call    exit@plt
```

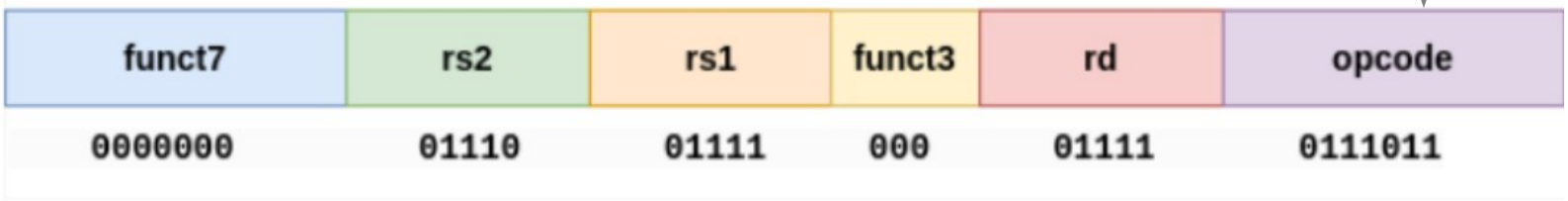
only add and shift

```
gcc MultTen.s -march=rv64g -o MultTen

riscv64-linux-gnu-objdump -d  MultTen
```

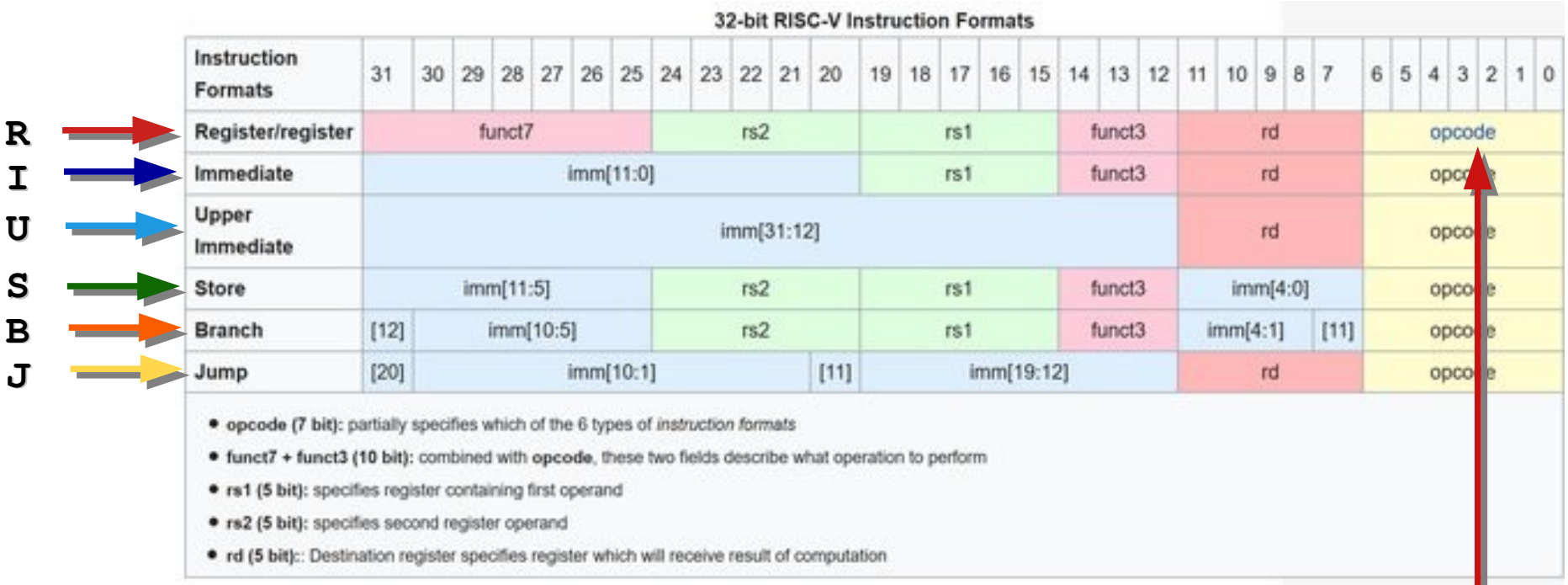
PLabs : arithmetics

```
00000000000006b8 <main>:          # virtual address - 24 bytes
 6b8:   00400793                  li      a5,4
 6bc:   00078713                  mv      a4,a5
 6c0:   0027979b                  sllw    a5,a5,0x2
 6c4:   00e787bb                  addw    a5,a5,a4
 6c8:   0017979b                  sllw    a5,a5,0x1
 6cc:   00078593                  mv      a1,a5
```

RV64I - R-type : 00e787bb addw a5,a5,a4

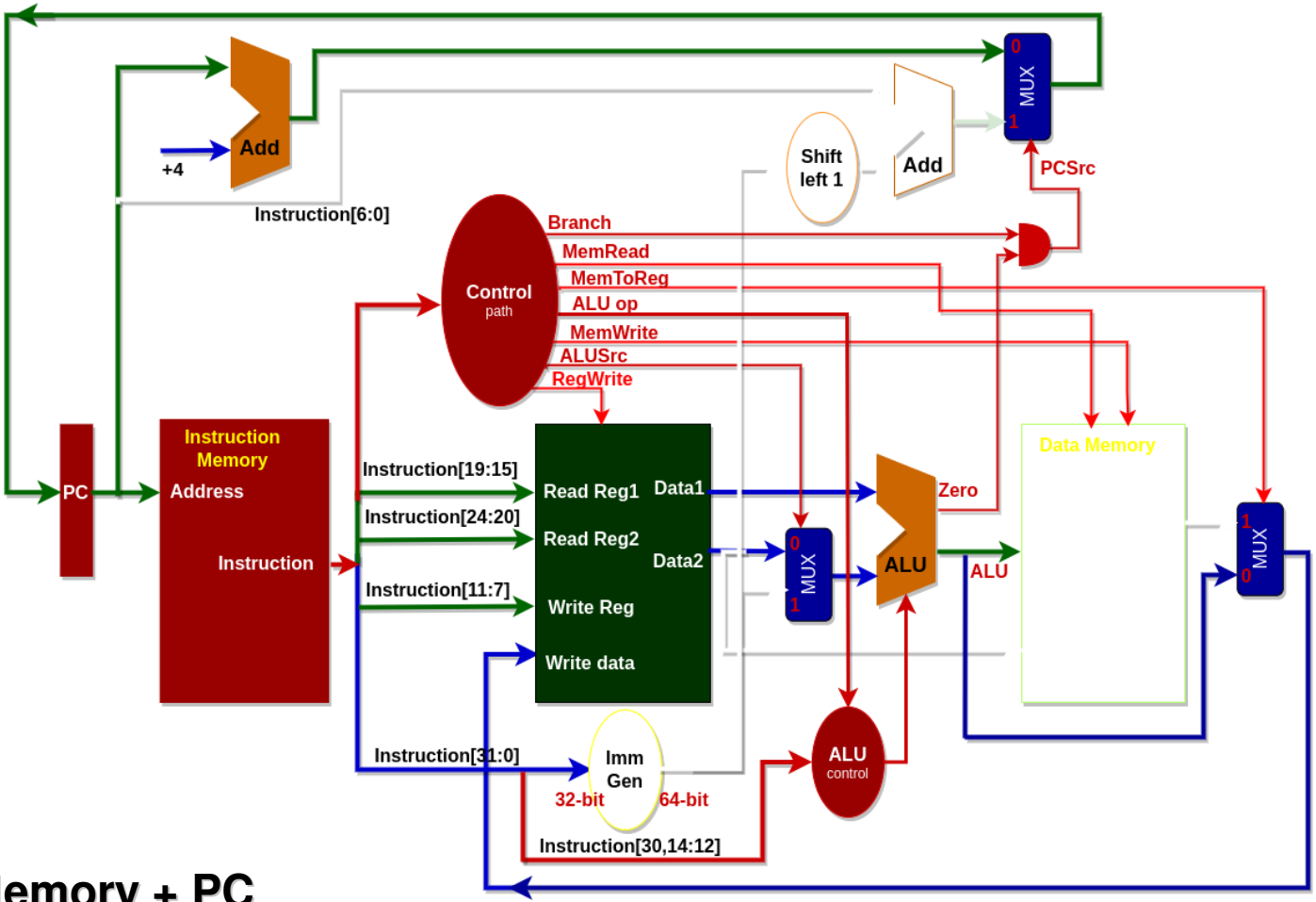


RV32I ISA - formats




```
addw a5,a5,a4
addw x15,x15,x14
```

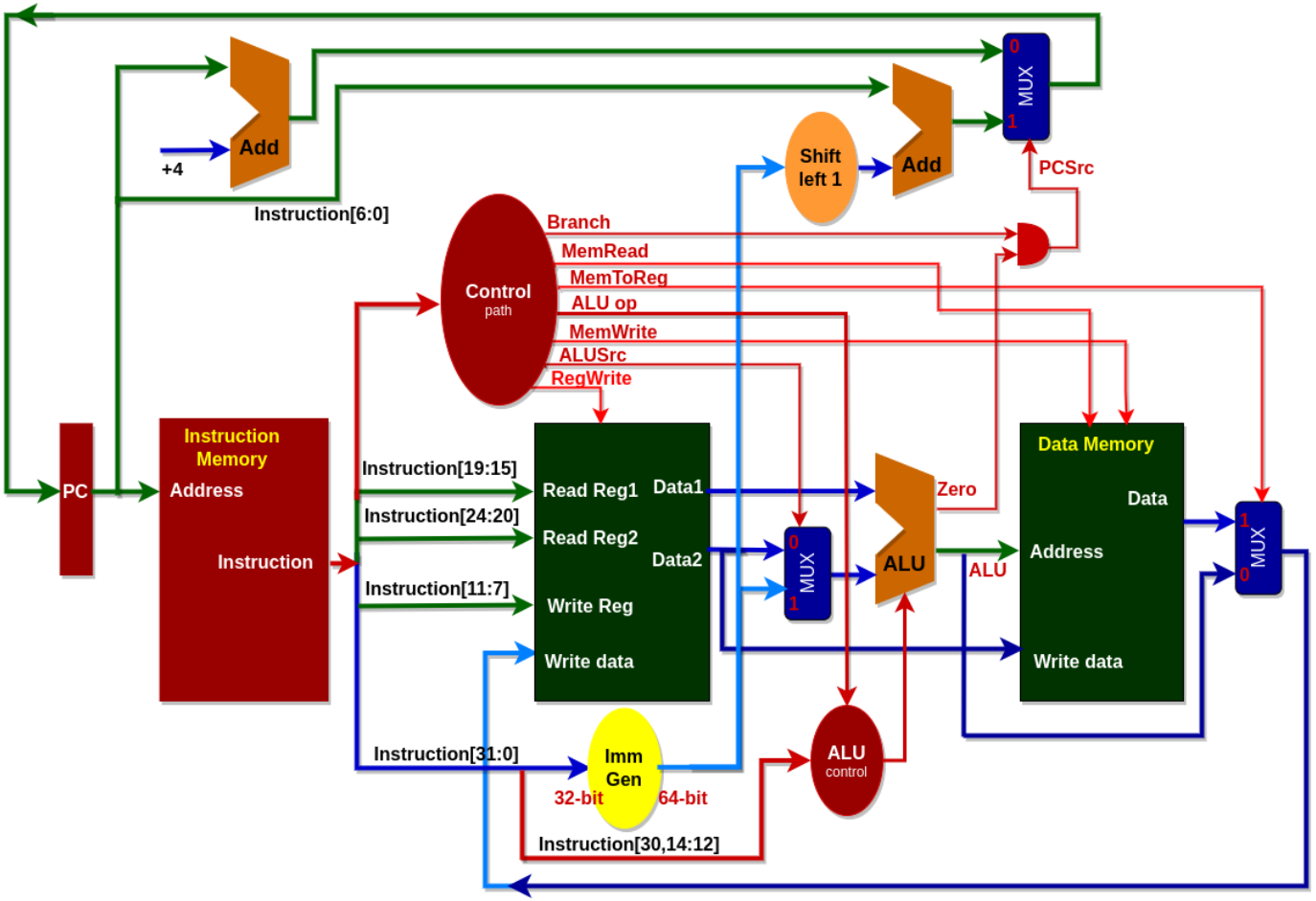
DLabs : Verilog (0-3) & Design (4-..)



DLab4

R-type only:
Instruction Memory + PC
Registers +ALU

DLab 5 : RV32I

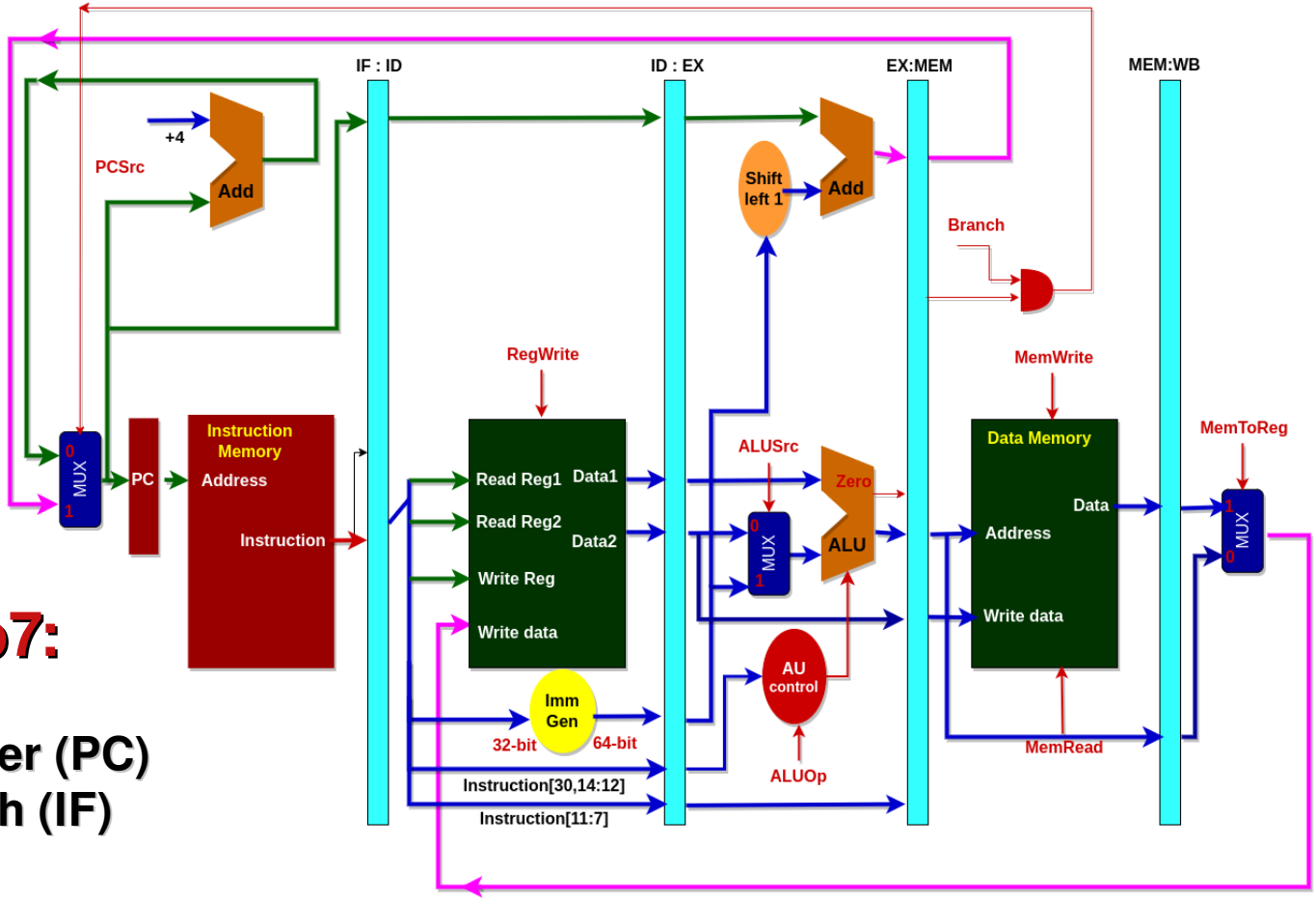


DLab5

RV32I

No pipeline

DLabs 6,7 .. : RV32I pipeline



DLab6, DLab7:

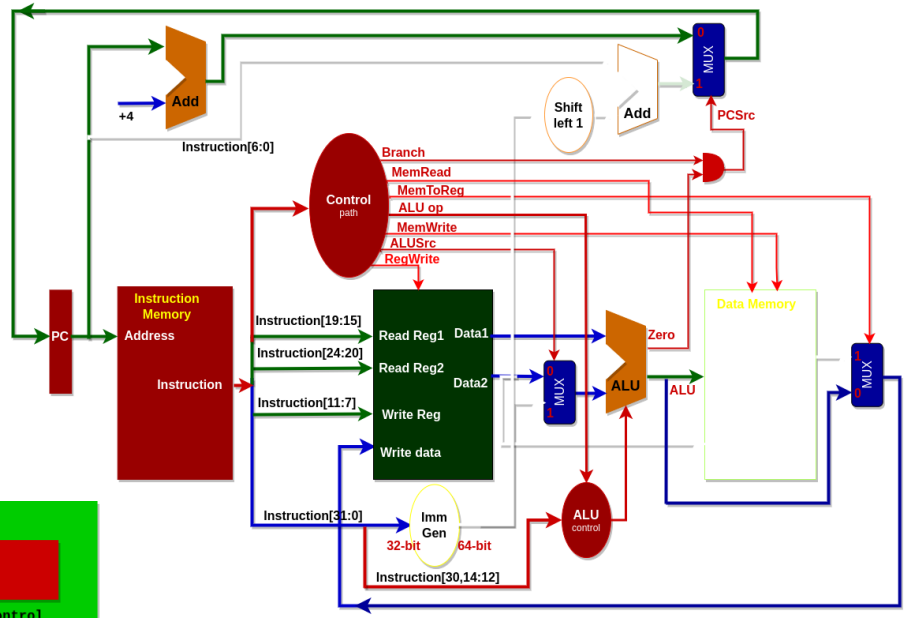
5-stage pipeline

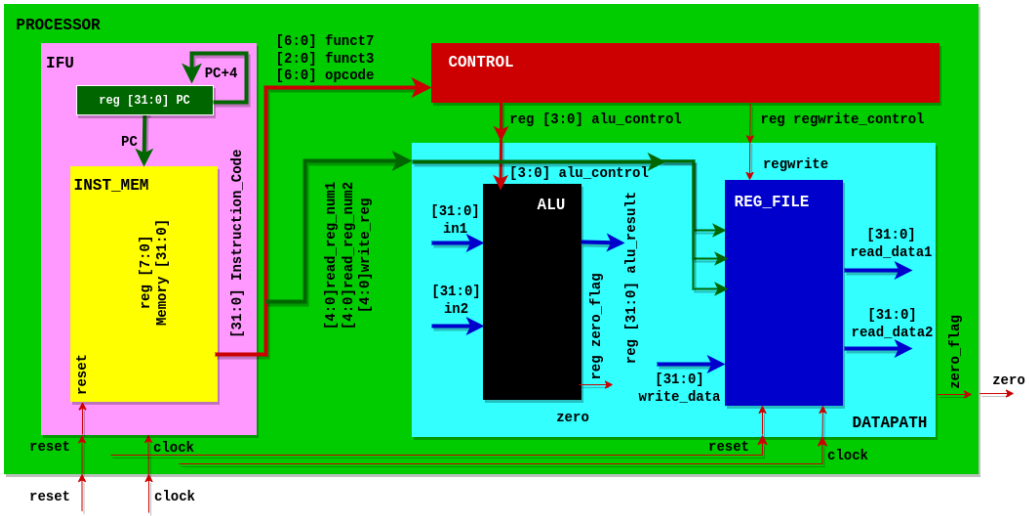
- Program Counter (PC)
- InstructionFetch (IF)
- DataRead (DR)
- Execute (EX)
- WriteBack (WB)

DLab4 : First design: R-type

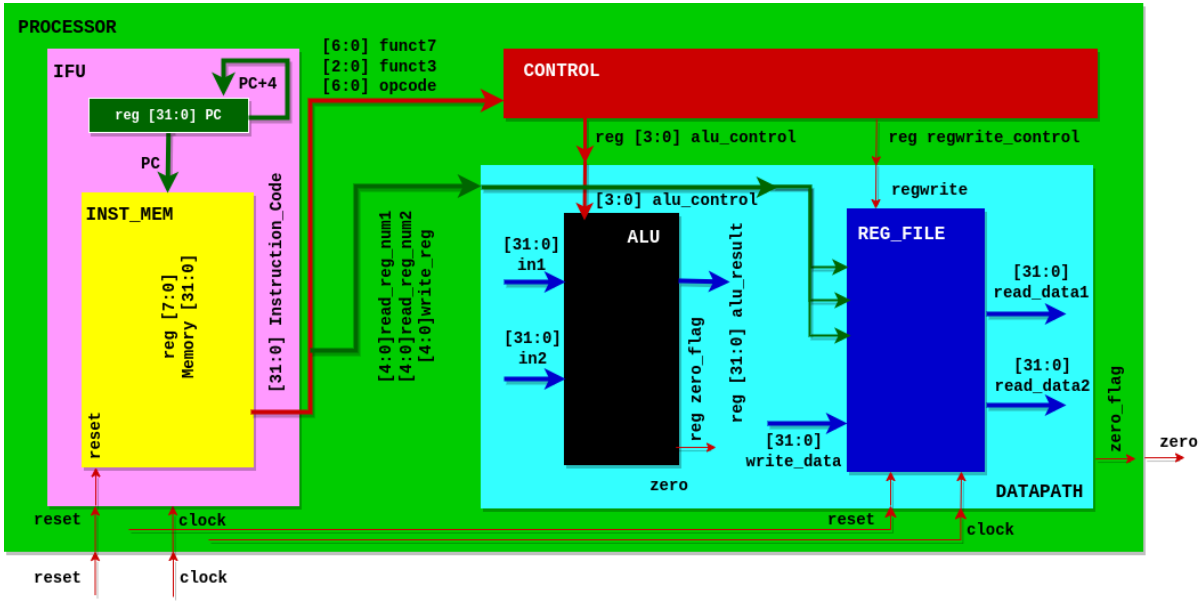
RVI - R-type : addw a5,a5,a4

funct7	rs2	rs1	fun3	rd	opcode
0000000	01110	01111	000	01111	0111011



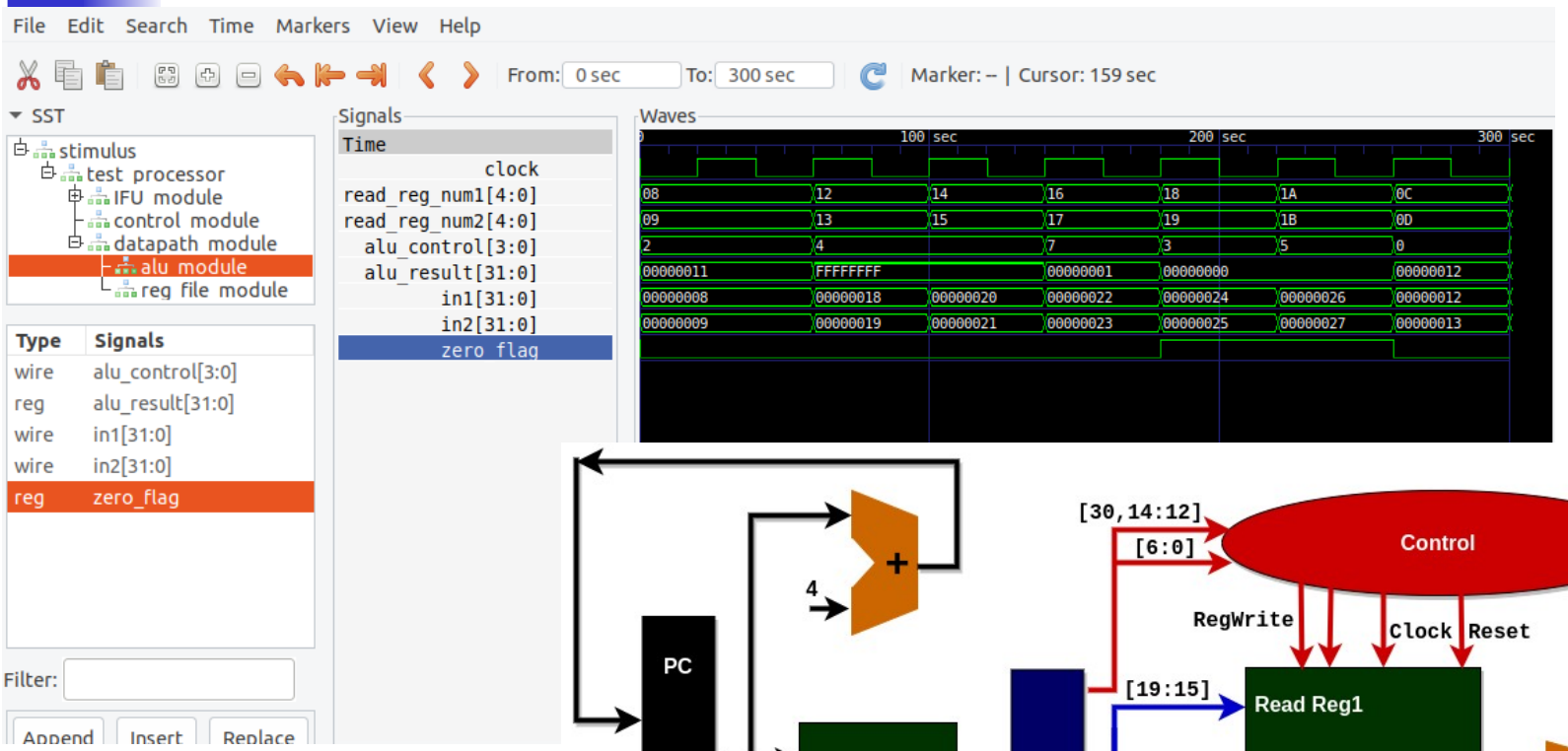


DLab4 : PROCESSOR

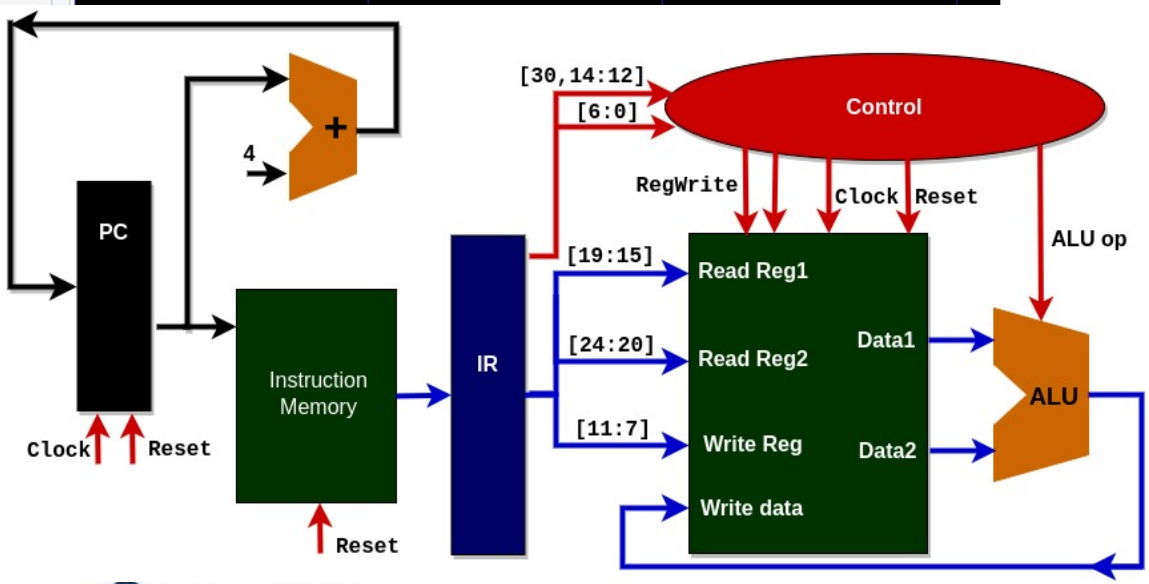



```
`include "CONTROL.v"
`include "DATAPATH.v"
`include "IFU.v"

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
```

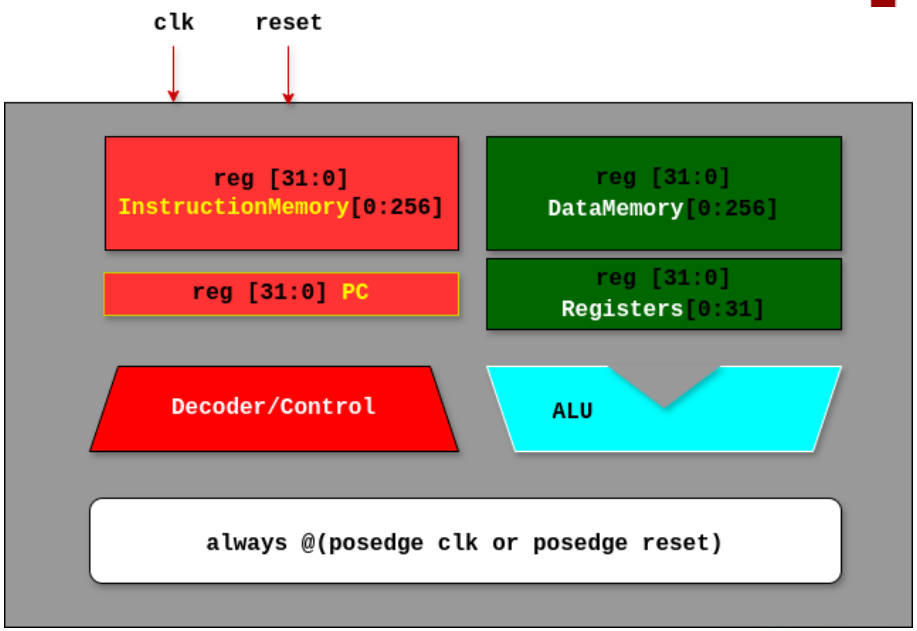
DLab4 : test & waveforms

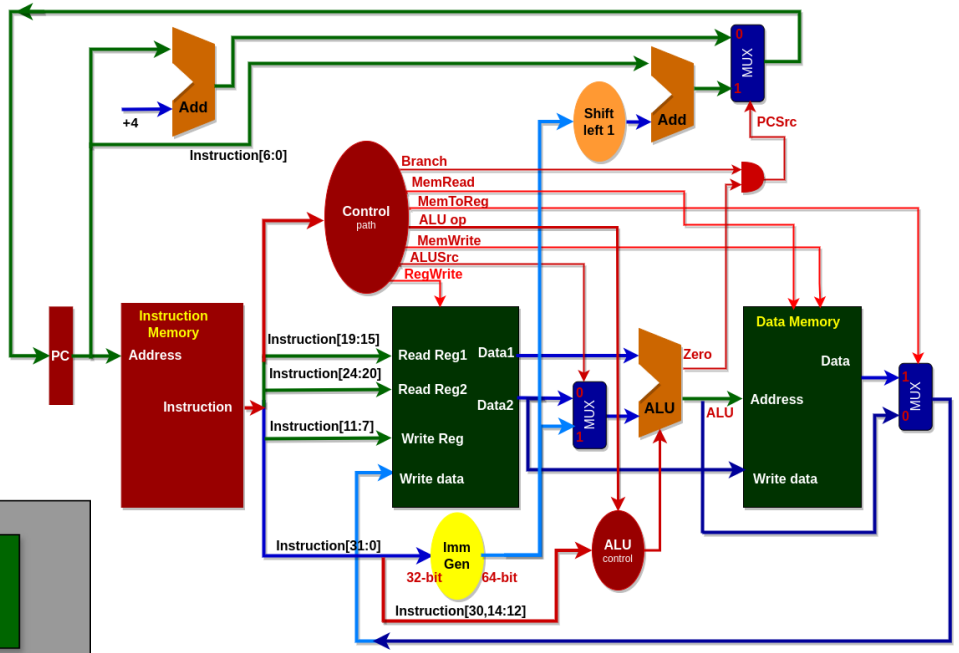


GTKWave waveforms



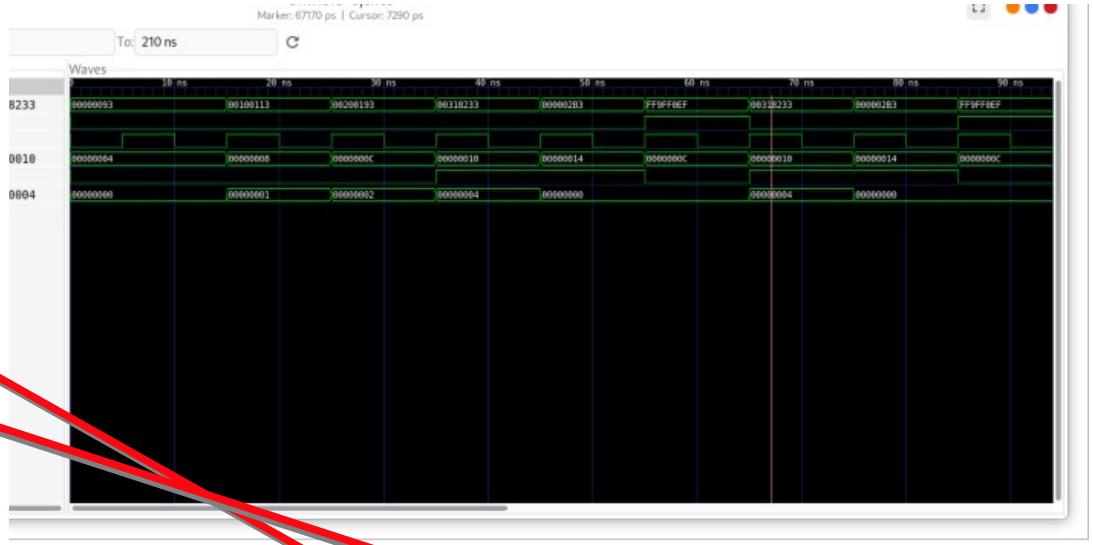
DLab5 : RV32I architecture





DLab5 : test and waveforms

```
00000093  // addi x1, x0, 0
00100113  // addi x2, x0, 1
00200193  // addi x3, x0, 2
00318233  // add x4, x3, x3
000002B3  // add x5, x0, x0
//008000EF  // jal x1, 8        - PC=PC+8 ,
FF9FF0EF  // jal x1, 8        - PC=PC-8 ,
00028263  // beq x5, x0, 8
00500293  // addi x5, x0, 5
00530333  // and x6, x0, x5
0330a023  // sw x1, 0(x3)
0670a023  // sw x1, 0(x3)
06f0a023  // lb x1, 0(x3)
04f0a023  // lbu x1, 0(x3)
0330a023  // sw x1, 0(x3)
0670a023  // sw x1, 0(x3)
06f0a023  // lb x1, 0(x3)
..
```



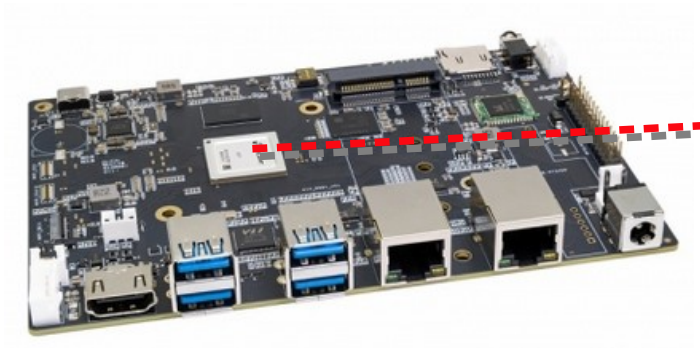
The Platform

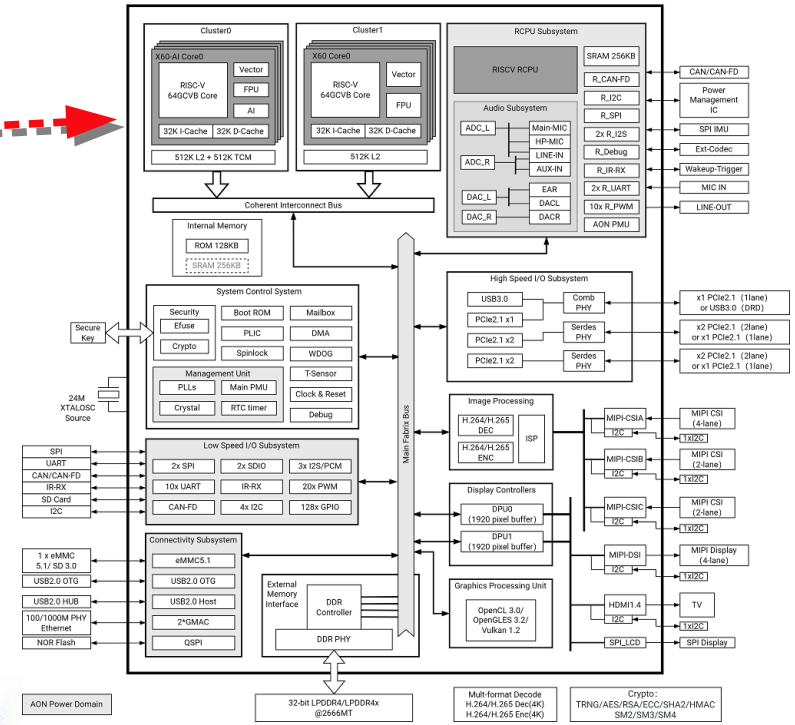
Software:

Ubuntu, C/V-intrinsics/assembly, `gcc`, `riscv64-linux-gnu-objdump`; Verilog, `iverilog`, `GTKwave`, ..

Hardware:

BPI-SF3, SpacemiT K1–X60: RV64GCVB + AI (16 instructions)





4*RV64GCVB + AI

and

4*RV64GCVB

The Platform

Thank you for your attention !

Source: Semico Research Corp.