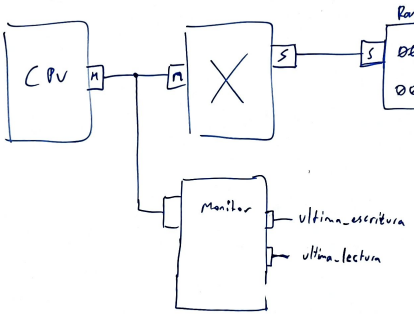
Test Bench




```
.data
val_a: .word 5
val_b: .word 4
resultado: .word 0
.section .text.init
.globl reset
reset:
   lw x1, val_a
   lw x2, val_b
   add x3, x1, x2
   sw x3, resultado, x4
fin:
   j fin
```

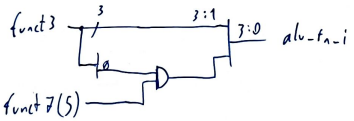
Modos ALU

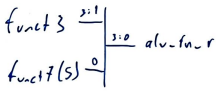
instrucciones con valor inmediato

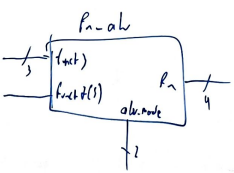
Op (decimal)		funct3	funct7(5)	fn_alu
19	addi	000	—	0000
	slli	001	0	0010
	slti	010	-	0100
	sltiu	011	-	0110
	xori	100	-	1000
	srli	101	0	1010
	srai	101	1	1011
	ori	110	-	1100
	andi	111	-	1110

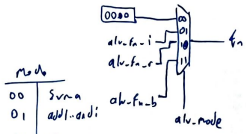
instrucciones entre registros

OP (decimal)		funct3	funct7(5)	fn_alu
51	add	000	0	0000
	sub	000	1	0001
	sll	001	0	0010
	slt	010	0	0100
	sltu	011	0	0110
	xor	100	0	1000
	srl	101	0	1010
	sra	101	1	1011
	or	110	0	1100
	and	111	0	1110







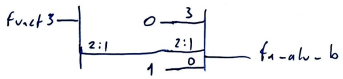


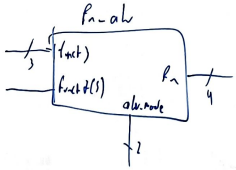
mode	
00	suma
01	addi-andi
10	add-and
11	6XX

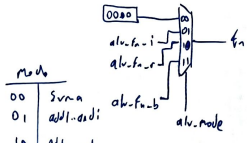
Modos ALU

Condiciones de salto

OP (decimal)		funct3	funct7(5)	alu_fn_b
99	beq	000	—	0001 | 100-
	bne	001	—	0001 | 100-
	blt	100	—	010-
	bge	101	—	010-
	bltu	110	—	011-
	bgeu	111	—	011-







mode	
00	Suma
01	addi_sub
10	add_and
11	bXX

Carga desde memoria

L{b | h | w | bu | hu} rd, imm(rs1)

[op=3]

addr := rs1 + imm

imm: valor inmediato
rs1: registro fuente
rd : registro destino

Lb : rd ← SignExt(bus(addr)(7 downto 0))
Lh : rd ← SignExt(bus(addr)(15 downto 0))
Lw : rd ← bus(addr)
Lbu : rd ← ZeroExt(bus(addr)(7 downto 0))
Lhu : rd ← ZeroExt (bus(addr)(15 downto 0))

por defecto
wpc ← 0 ;
wmem ← 0 ;
winst ← 0 ;
wreg ← 0 ;
jump ← 0 ;
sipc ← 0 ;
alu_mode ← 00 ;
imm_mode ← 000 ;
sel_imm ← 0 ;
data_addr ← 0 ;
mem_source ← 0 ;

Carga la instrucción (fetch)

T1 : data_addr ← 0;

T2 : winst ← 1 ;

Carga rs1 y avanza PC — carga la instrucción

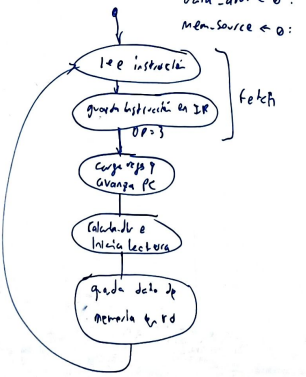
T3 : wpc ← 1;

Calcula la dirección y pide el dato

T4: alu_mode ← 00 : sel_imm ← 1 : data_addr ← 1;

guarda el dato en rd

T5: mem_source ← 1 : wreg ← 1;



Carga desde memoria

L{b | h | w | bu | hu} rd, imm(rs1)

[op=3]

addr := rs1 + imm

imm: valor inmediato
rs1: registro fuente
rd : registro destino

L b : rd ← SignExt(bus(addr)(7 downto 0))
L h : rd ← SignExt(bus(addr)(15 downto 0))
L w : rd ← bus(addr)
L bu : rd ← ZeroExt(bus(addr)(7 downto 0))
L hu : rd ← Zero Ext (bus(addr)(15 downto 0))

por defecto
wpc ← 0 ;
wmem ← 0 ;
wirst ← 0 ;
wreg ← 0 ;
jump ← 0 ;
sipc ← 0 ;
alu_mode ← 00 ;
imm_mode ← 000 ;
sel_imm ← 0 ;
data_addr ← 0 ;
mem_source ← 0 ;

Carga la instrucción (fetch)

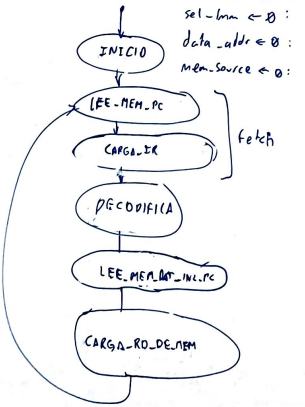
T1 : data_addr ← 0;
T2 : wirst ← 1 ;
Decodifica
T3 : ;

Calcula la dirección y pide el dato, inc. PC (cargo la instrucción y tengo rs1 y rs2)

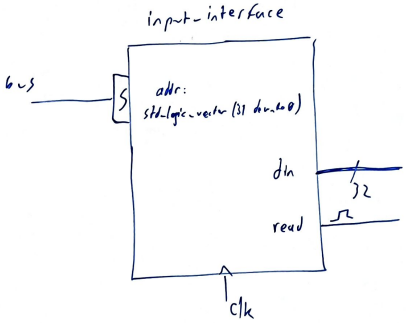
T4: alu_mode ← 00 : sel_imm ← 1 : data_addr ← 1 : wpc ;

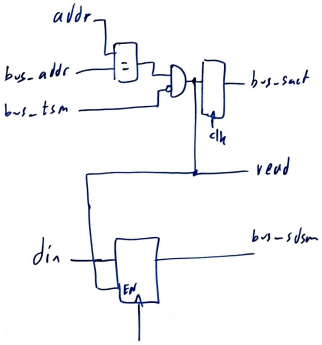
guarda el dato en rd

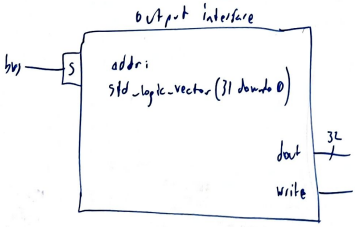
T5: mem_source ← 1 : wreg ← 1 ;

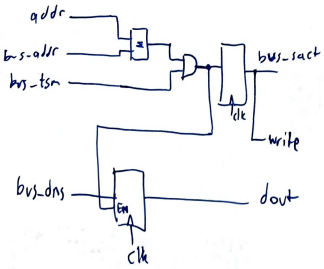


bus_saddr
bus_sdms
bus_stms
bus_stwidth
bus_sact
bus_sdsm









bus_saddr
bus_sdms
bus_stms
bus_stwidth
bus_sact
bus_sdsm