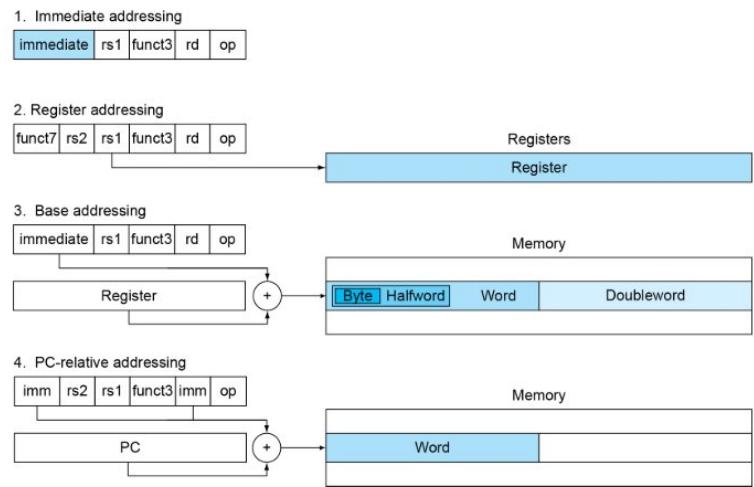


# RISCV\_LITE\_DESIGN

Category	Name	Fmt	RV32I Base
Loads	Load Byte	I	LB rd,rs1,imm
	Load Halfword	I	LH rd,rs1,imm
	Load Word	I	LW rd,rs1,imm
	Load Byte Unsigned	I	LBU rd,rs1,imm
	Load Half Unsigned	I	LHU rd,rs1,imm
Stores	Store Byte	S	SB rs1,rs2,imm
	Store Halfword	S	SH rs1,rs2,imm
	Store Word	S	SW rs1,rs2,imm
Shifts	Shift Left	R	SLL rd,rs1,rs2
	Shift Left Immediate	I	SLLI rd,rs1,shamt
	Shift Right	R	SRL rd,rs1,rs2
	Shift Right Immediate	I	SRLI rd,rs1,shamt
	Shift Right Arithmetic	R	SRA rd,rs1,rs2
	Shift Right Arith Imm	I	SRAI rd,rs1,shamt
Arithmetic	ADD	R	ADD rd,rs1,rs2
	ADD Immediate	I	ADDI rd,rs1,imm
	SUBtract	R	SUB rd,rs1,rs2
	Load Upper Imm	U	LUI rd,imm
	Add Upper Imm to PC	U	AUIPC rd,imm
Logical	XOR	R	XOR rd,rs1,rs2
	XOR Immediate	I	XORI rd,rs1,imm
	OR	R	OR rd,rs1,rs2
	OR Immediate	I	ORI rd,rs1,imm
	AND	R	AND rd,rs1,rs2
	AND Immediate	I	ANDI rd,rs1,imm
Compare	Set <	R	SLT rd,rs1,rs2
	Set < Immediate	I	SLTI rd,rs1,imm
	Set < Unsigned	R	SLTU rd,rs1,rs2
	Set < Imm Unsigned	I	SLTIU rd,rs1,imm
Branches	Branch =	SB	BEQ rs1,rs2,imm
	Branch ≠	SB	BNE rs1,rs2,imm
	Branch <	SB	BLT rs1,rs2,imm
	Branch ≥	SB	BGE rs1,rs2,imm
	Branch < Unsigned	SB	BLTU rs1,rs2,imm
	Branch ≥ Unsigned	SB	BGEU rs1,rs2,imm
Jump & Link	J&L	UJ	JAL rd,imm
	Jump & Link Register	UJ	JALR rd,rs1,imm



Name (Field Size)	Field						Comments
	7 bits	5 bits	5 bits	3 bits	5 bits	7 bits	
R-type	funct7	rs2	rs1	funct3	rd	opcode	Arithmetic instruction format
I-type	immediate[11:0]		rs1	funct3	rd	opcode	Loads & immediate arithmetic
S-type	immed[11:5]	rs2	rs1	funct3	immed[4:0]	opcode	Stores
SB-type	immed[12,10:5]	rs2	rs1	funct3	immed[4:1,11]	opcode	Conditional branch format
UJ-type	immediate[20,10:1,11,19:12]				rd	opcode	Unconditional jump format
U-type	immediate[31:12]				rd	opcode	Upper immediate format

In verde sono evidenziate tutte le istruzioni RISC-V da implementare per eseguire "minv.exe" (14 istruzioni differenti).

Nel codice assembly invece i jump dovuti a chiamate a funzione sono le frecce verdi, mentre i jump dovuti a semplici branch condizionali sono le frecce rosse.

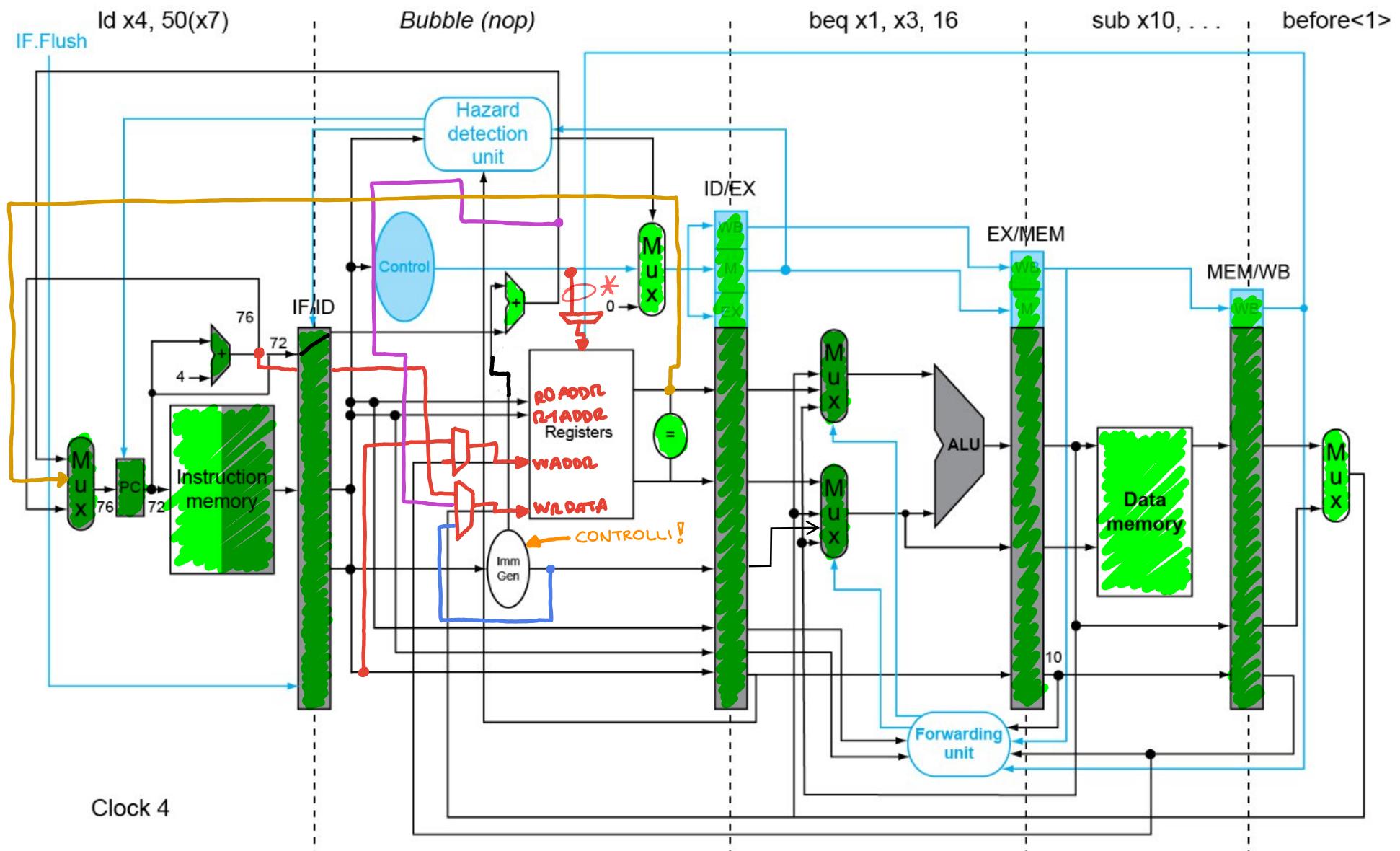
Tutte le istruzioni sono separate nella prima parte "INIT", che viene eseguita prima di entrare nel main, nella seconda parte "MAIN", e nella testa parte "MINV" che rappresenta le istruzioni interne alla funzione minv().

Address	Code	Basic	Source
0x00400000	0x1fc18197	auipc x3,0x0001fc18	16: auipc gp,0x1fc18
0x00400004	0x01c18193	addi x3,x3,28	17: addi gp,sp,28
0x00400008	0x7fbff117	auipc x2,0x0007fbff	18: auipc sp,0x7fbff
0x0040000c	0xff010113	addi x2,x2,0xffffffff4	19: addi sp,sp,-12
0x00400010	0x00010433	addi x8,x2,x0	20: add s0,sp,zero
0x00400014	0x008000ef	jal x1,0x00000008	21: jal ra,main
0x00400018	0x0000006f	jal x0,0x00000000	24: j el
0x0040001c	0x10010537	lui x10,0x00010001	27: lui a0,0x10010 <main> JMP
0x00400020	0xff010113	addi x2,x2,0xffffffff0	28: addi sp,sp,-16
0x00400024	0x00700593	addi x11,x0,7	29: li a1,7
0x00400028	0x00a00533	add x10,x0,x10	30: mv a0,a0
0x0040002c	0x00112623	sw x1,12(x2)	31: sw ra,12(sp)
0x00400030	0x01c000ef	jal x1,0x0000001c	32: jal ra,minv
0x00400034	0x00c12083	lw x1,12(x2)	33: lw ra,12(sp)
0x00400038	0x100107b7	lui x15,0x00010010	34: lui a5,0x10010
0x0040003c	0x00a7ee23	sw x10,28(x15)	35: sw a0,28(a5)
0x00400040	0x00000513	addi x10,x0,0	36: li a0,0
0x00400044	0x01010113	addi x2,x2,16	37: addi sp,sp,16
0x00400048	0x00000867	jalr x0,x1,0	38: ret
0x0040004c	0x000052603	lw x12,0(x10)	41: lw a2,0(a0) <minv> JMP
0x00400050	0x00100713	addi x14,x0,1	42: li a4,1
0x00400054	0x41f65793	srai x15,x12,31	43: srai a5,a2,0x1f
0x00400058	0x000c7c633	xor x12,x15,x12	44: xor a2,a5,a2
0x0040005c	0x40f60633	sub x12,x12,x15	45: sub a2,a2,a5
0x00400060	0x02b75863	bge x14,x11,0x00000008	46: ble a1,a4,1bl2
0x00400064	0x00259593	sll1 x11,x11,x11,2	47: slli a1,a1,0x2
0x00400068	0x00450713	addi x14,x10,4	48: addi a4,a0,4
0x0040006c	0x000b50533	add x10,x10,x11	49: add a0,a0,a1
0x00400070	0x00072783	lw x15,0(x14)	52: lw a5,0(a4) <lbl0>
0x00400074	0x00470713	addi x14,x14,4	53: addi a4,a4,4
0x00400078	0x41f7d4693	srai x13,x15,31	54: srai a3,a5,0x1f
0x0040007c	0x00f6c7b3	xor x15,x13,x15	55: xor a5,a3,a5
0x00400080	0x40d787b3	sub x15,x15,x13	56: sub a5,a5,a3
0x00400084	0x00c7d463	bge x15,x12,0x00000008	57: ble a2,a5,1bl1
0x00400088	0x00f00633	add x12,x0,x15	58: mv a2,a5
0x0040008c	0xfee512e3	one x10,x14,0xffffffff4	61: bne a0,a4,1bl0 <lbl1>
0x00400090	0x00c00533	add x10,x0,x12	64: mv a0,a2 <lbl2>
0x00400094	0x00000867	jalr x0,x1,0	65: ret

In verde sono evidenziati i blocchi già realizzati (solo da modificare).

Hazard detection unit, forwarding unit, control unit e immediate gen sono tutti blocchi combinatori.

Per i vari segnali di comando, per il campo opcode delle istruzioni, valutare l'utilizzo di segnali enumerate per astrarre dalla codifica dei segnali di controllo e interpretare facilmente le operazioni durante la simulazione.



## MODIFICHE STRUTTURALI

Le istruzioni che richiedono delle modifiche al sistema sono:

- JAL: modifiche riportate in rosso
- JALR: modifiche riportate in giallo
- LUI: modifiche riportate in blu
- AUIPC: modifiche riportate in viola

Le istruzioni restanti sono già compatibili con l'architettura.

## NOTE SU ASTERISCO SOPRA REGISTER FILE

I due ingressi al multiplexer rappresentano entrambi il write enable del register file: questi due segnali devono avere due uscite differenti dall'unità di controllo, se l'uscita è la stessa le modifiche fatte al sistema non funzionano.

## REGISTRO x0

- x0 non va sovrascritto, è fisso al valore '0'
- l'istruzione "jalr" potrebbe sovrascrivere x0 durante "ret"

## DA FARE

- gestire i vari segnali di controllo per tutti i componenti
  - ricordare che per le istruzioni di branch (BEQ, BNE...) è necessario valutare il risultato del confronto tra due registri, il risultato è fornito da un opportuno segnale ("zero" o altri segnali), con il quale si decide come pilotare il mux all'ingresso del PC (quindi è necessario un blocco di controllo apposta per questo mux)
  - valutare come definire i tipi enumerate nel sistema
- valutare il parallelismo degli operandi e quali vanno definiti signed
- realizzare un meccanismo strutturale per rendere il registro x0 read-only

