Hazard Detection Unit

HDU

PREFETCH ENABLE

FLUSH IR

IF ENABLE

ID ENABLE

EX ENABLE

ME ENABLE

CWs

CW2

CW3

CW4

4

+

PC

Instruction Memory

IRAM

NOP

NOP

IR

NPC

NPC (ID)

NPC (EX)

NPC (ME)

RS1

RS2

DATAIN

ADDR

Register File

RF

A

B

00

01

11

10

BZ

0?

MUX_COND_SEL

Branch Logic

COND

COND

ALU

ALU OUT

B (EX)

ADDR

DATAIN

Data Memory

DRAM

MEM LMD

LMD

ALU OUT (ME)

LMD

Control Unit

CU

INS

IMM

UNSIGNED EXTENDER

SIGNED

SIGN EXTENDER

J

IMM

J_IMM

SIGN EXTENDER

EX LMD

RS2

$LR

R

RD

RD

RD (EX)

RD (ME)

RT

RS

RT

RS

Forwarding Unit

FU

Legend
General Signals
Control Unit Signals
Hazard Detection Unit Signals
Forwarding Unit Signals (Data)
Forwarding Unit Signals (Control)
Stage Registers

FETCH

DECODE

EXECUTE

MEMORY

WRITEBACK

Control Unit

CU

Hazard Detection Unit

PREFETCH EN

FLUSH IR

IF ENABLE

ID ENABLE

EX ENABLE

ME ENABLE

CWs

CW2

CW3

CW4

1

+

PC

MUXJUMP

non c'è questo in violetto, permetterebbe di ridurre lo stallo della jump di 1

Instruction Memory

IRAM

NOP

NOP MUX

IR

NPC

NPC (ID)

NPC (EX)

NPC (ME)

RS1

RS2

DATAIN

ADDR

Register File

RF

A

B

IMM

RD

RT

RS

SIGN EXTENDER AND SHIFTER

UNSIGNED EXTENDER

SIGN EXTENDER

SIGN EXTENDER

IMM

J_IMM

MUX SIGN

MUX J

RS2

$LR

RD

MUX R

RT (secondo operando)
(valutare se va bene riusare RD)

RS (primo operando)

00

11

10

MUX_FWD_BZ

== 0?

MUX_COND_SEL

BEQZ/BNEZ comb. logic

COND

MUX COND

00

01

11

10

MUX A

31:2

MUX_JR

ALU

00

01

11

10

MUX B

MUX_FWD_EX_LMD

ALU OUT

B (EX)

RD (EX)

ADDR

DATAIN

MUX_FWD_MEM_LMD

Data Memory

DRAM

LMD

ALU OUT (ME)

RD (ME)

MUX LMD

Forwading unit

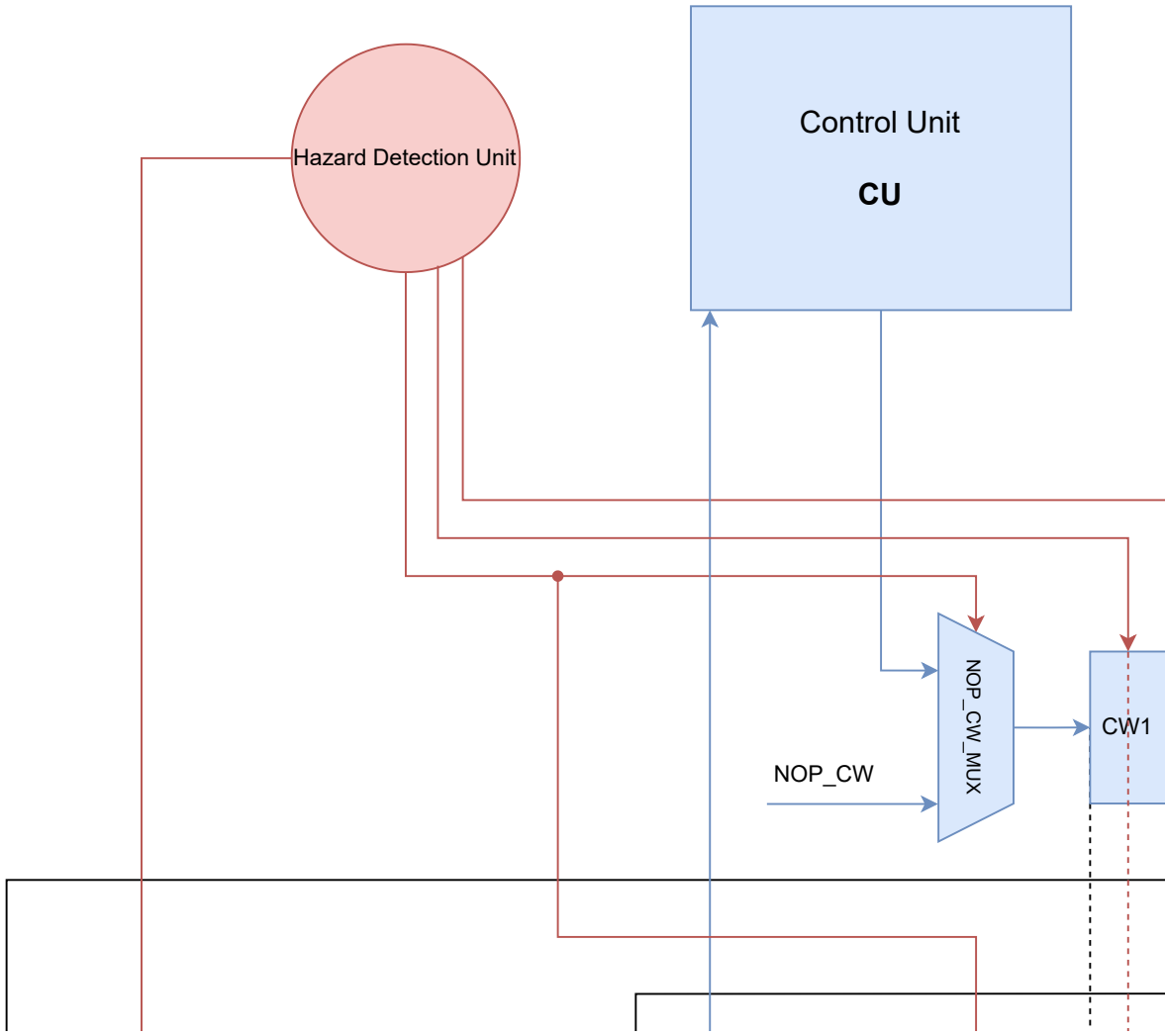
RD da EX

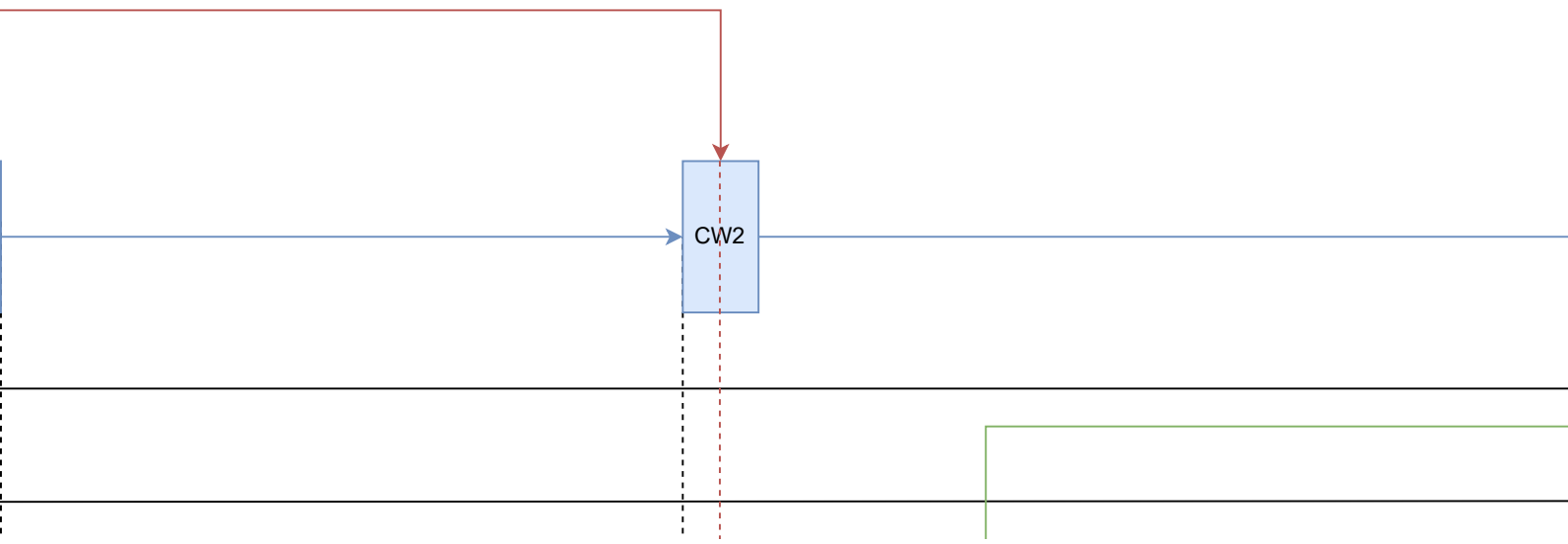
RD da MEM

Control Unit

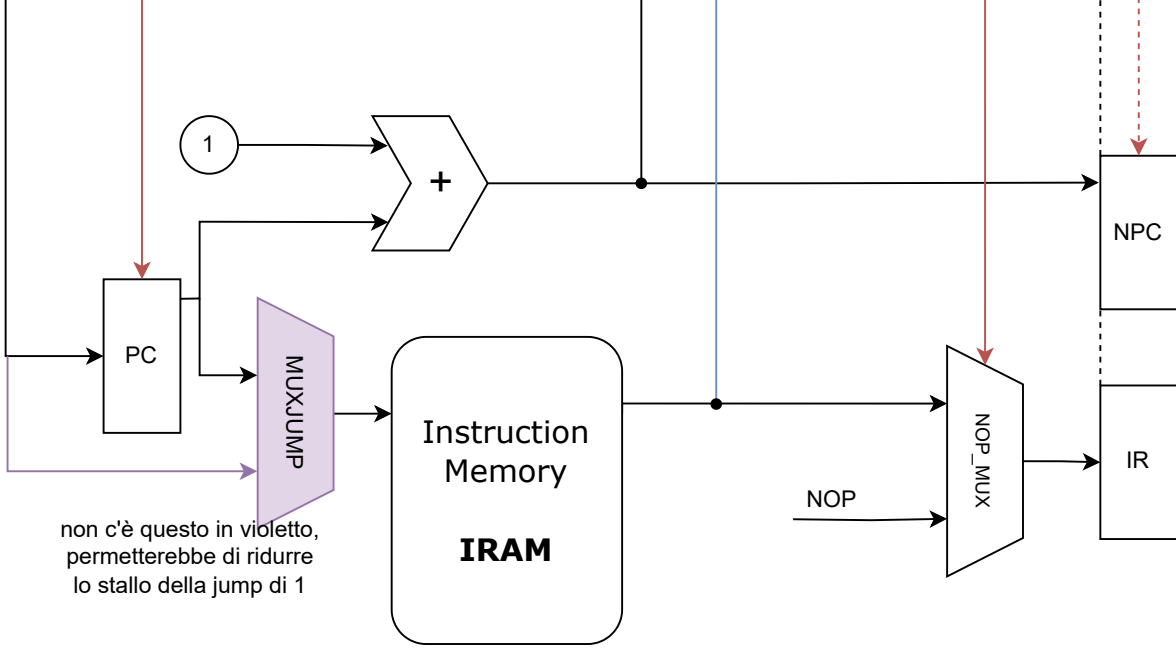
CU

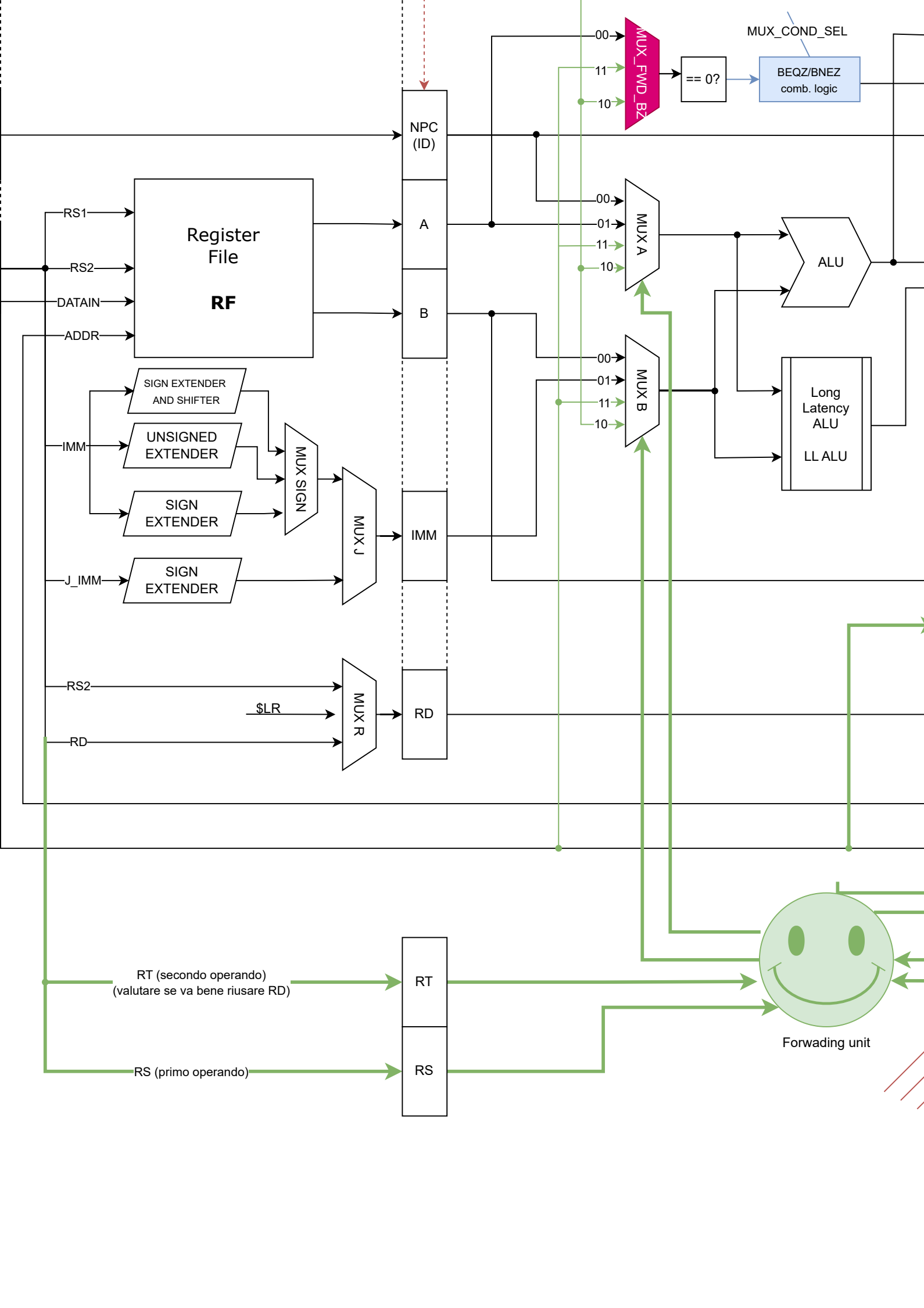
INS

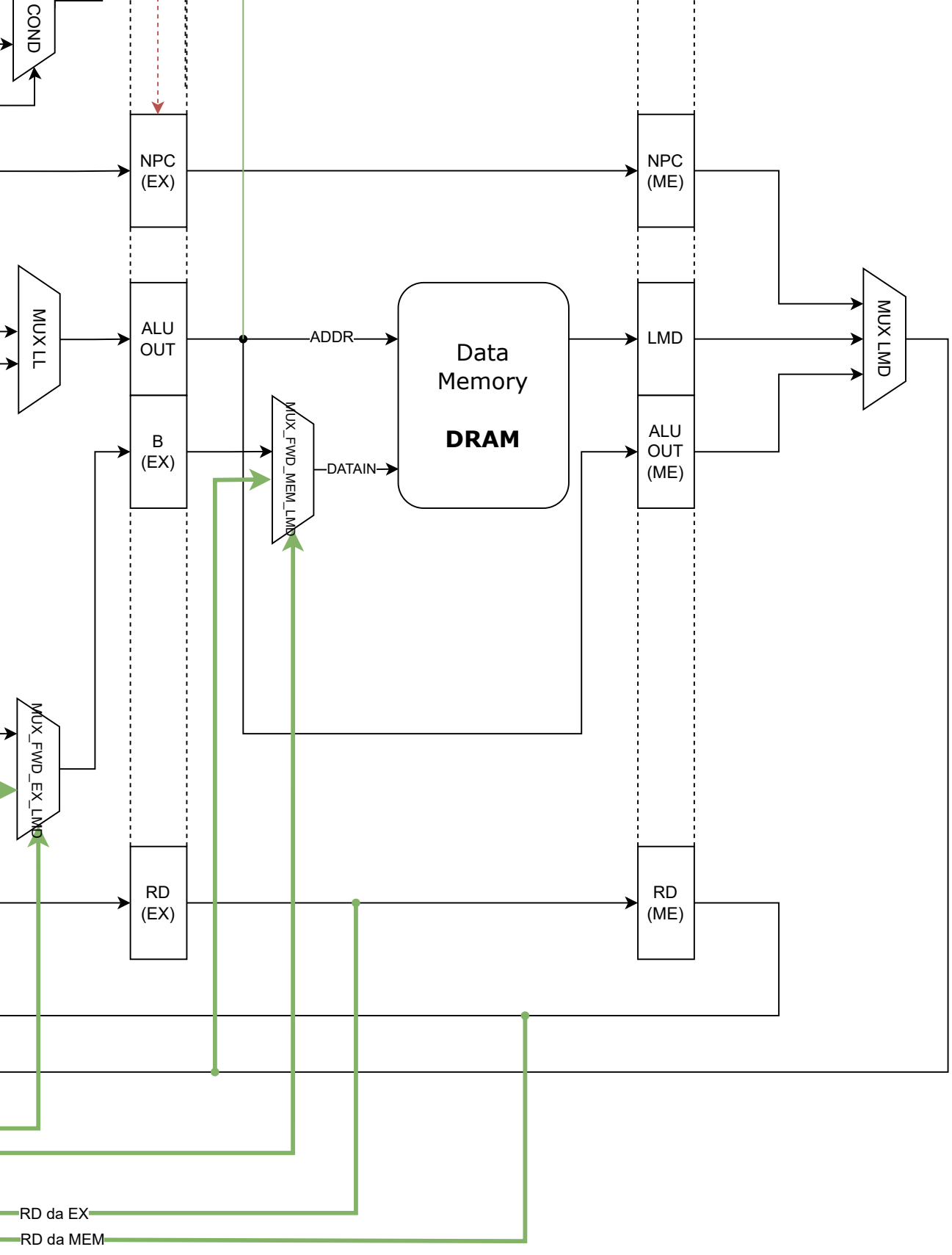

Hazard Detection Unit
Control Unit
CU
NOP_CW_MUX
NOP_CW
CW1

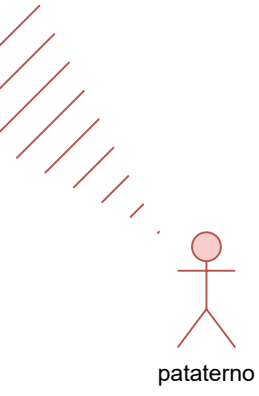

CW2

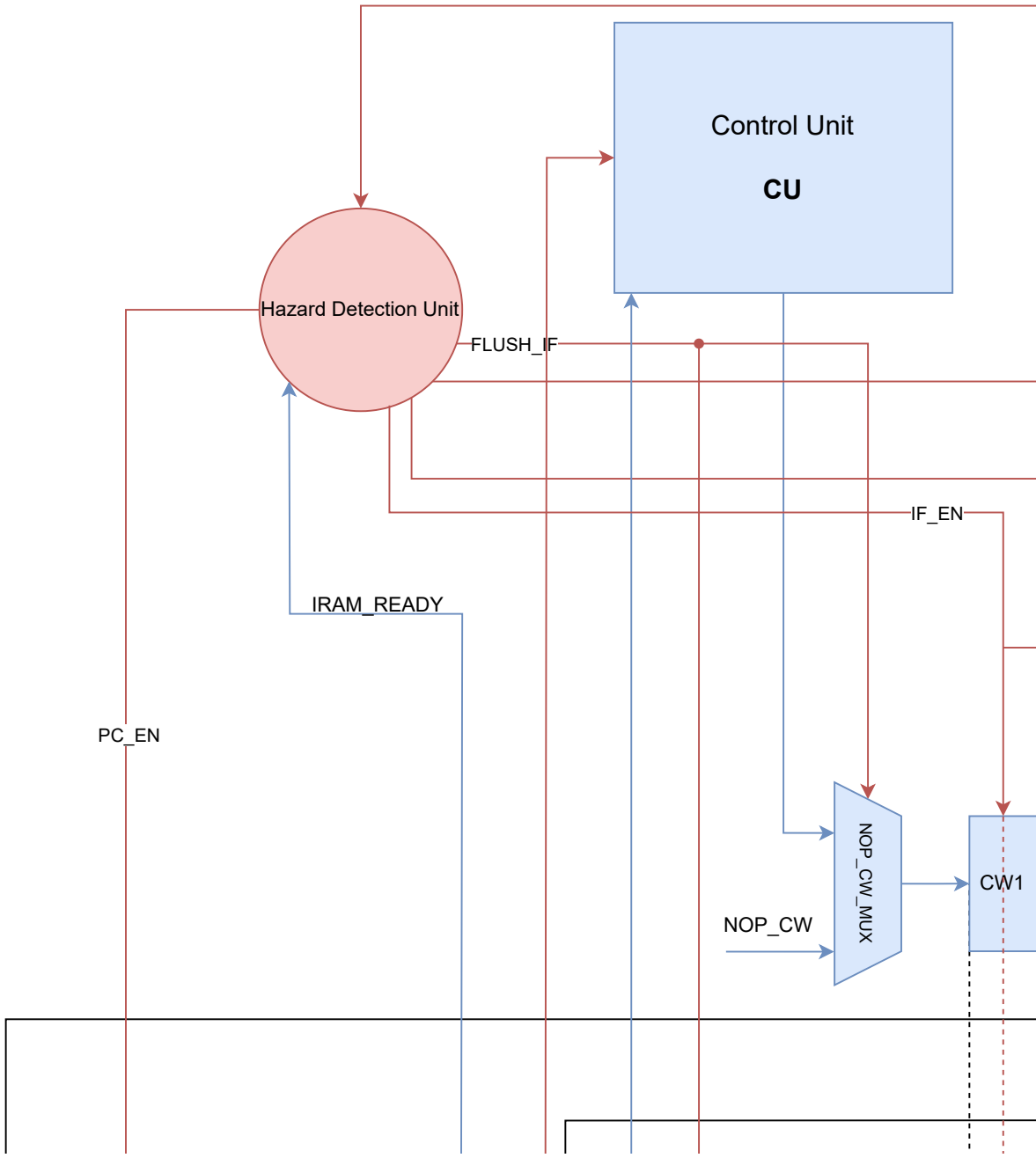

CW3
CW4
MUX


1
+
NPC
PC
MUXJUMP
Instruction
Memory
IRAM
NOP_MUX
NOP
IR
non c'è questo in violetto,
permetterebbe di ridurre
lo stallo della jump di 1


MUX_COND_SEL
00
11
10
MUX_FWD_BZ
== 0?
BEQZ/BNEZ
comb. logic
NPC
(ID)
RS1
RS2
DATAIN
ADDR
Register
File
RF
A
B
00
01
11
10
MUX A
ALU
00
01
11
10
MUX B
Long
Latency
ALU
LL ALU
SIGN EXTENDER
AND SHIFTER
IMM
UNSIGNED
EXTENDER
SIGN
EXTENDER
MUX SIGN
MUX J
IMM
J_IMM
SIGN
EXTENDER
RS2
$LR
RD
MUX R
RD
RT (secondo operando)
(valutare se va bene riusare RD)
RT
RS (primo operando)
RS
Forwading unit


COND
NPC (EX)
NPC (ME)
MUX LL
ALU OUT
ADDR
Data Memory
DRAM
LMD
MUX LMD
B (EX)
MUX_FWD_MEM_LMD
DATAIN
ALU OUT (ME)
MUX_FWD_EX_LMD
RD (EX)
RD (ME)
RD da EX
RD da MEM


pataterno


Control Unit
CU
Hazard Detection Unit
FLUSH_IF
IF_EN
IRAM_READY
PC_EN
NOP_CW_MUX
NOP_CW
CW1

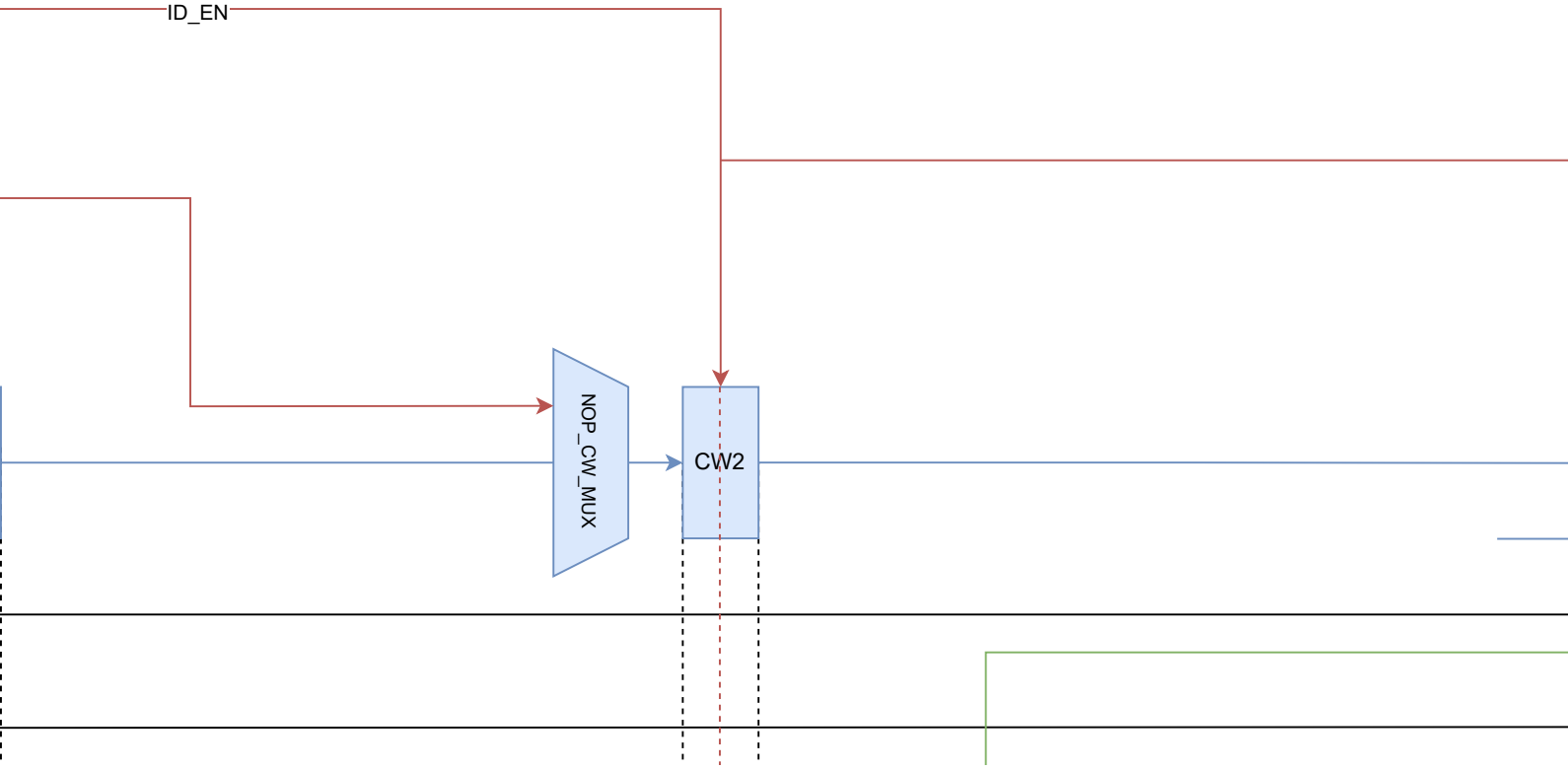
DRAM_REDY

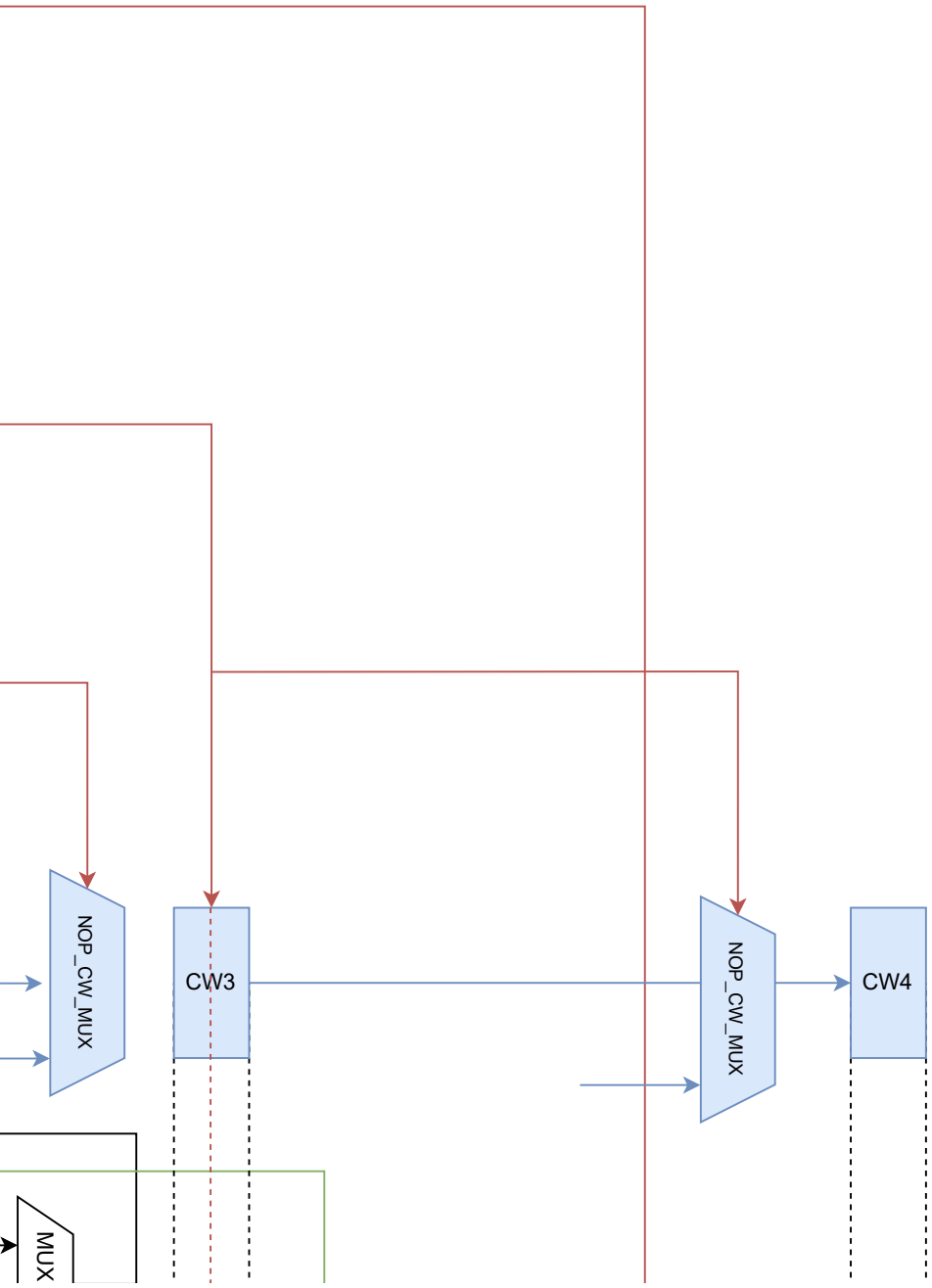
SEGNALE DIRETTAMENTE
DA IRAM, O passa dalla cu?

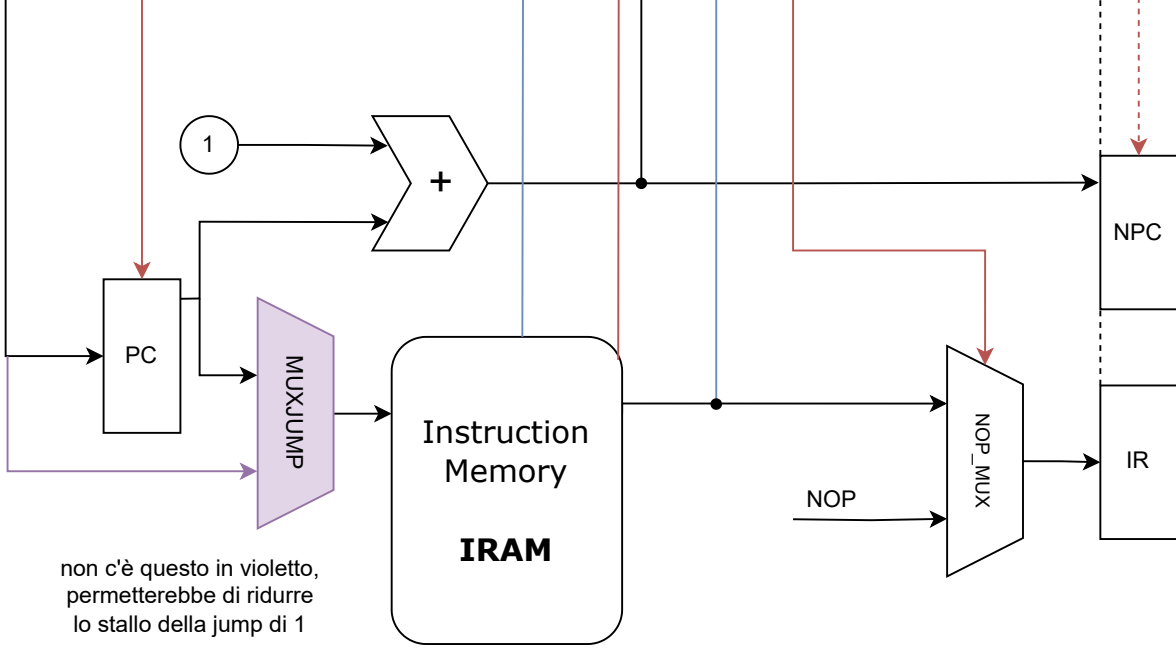
devo disabilitare == prevenire che vengano aggiornati
pc
if/id
id/exe
exe/mem

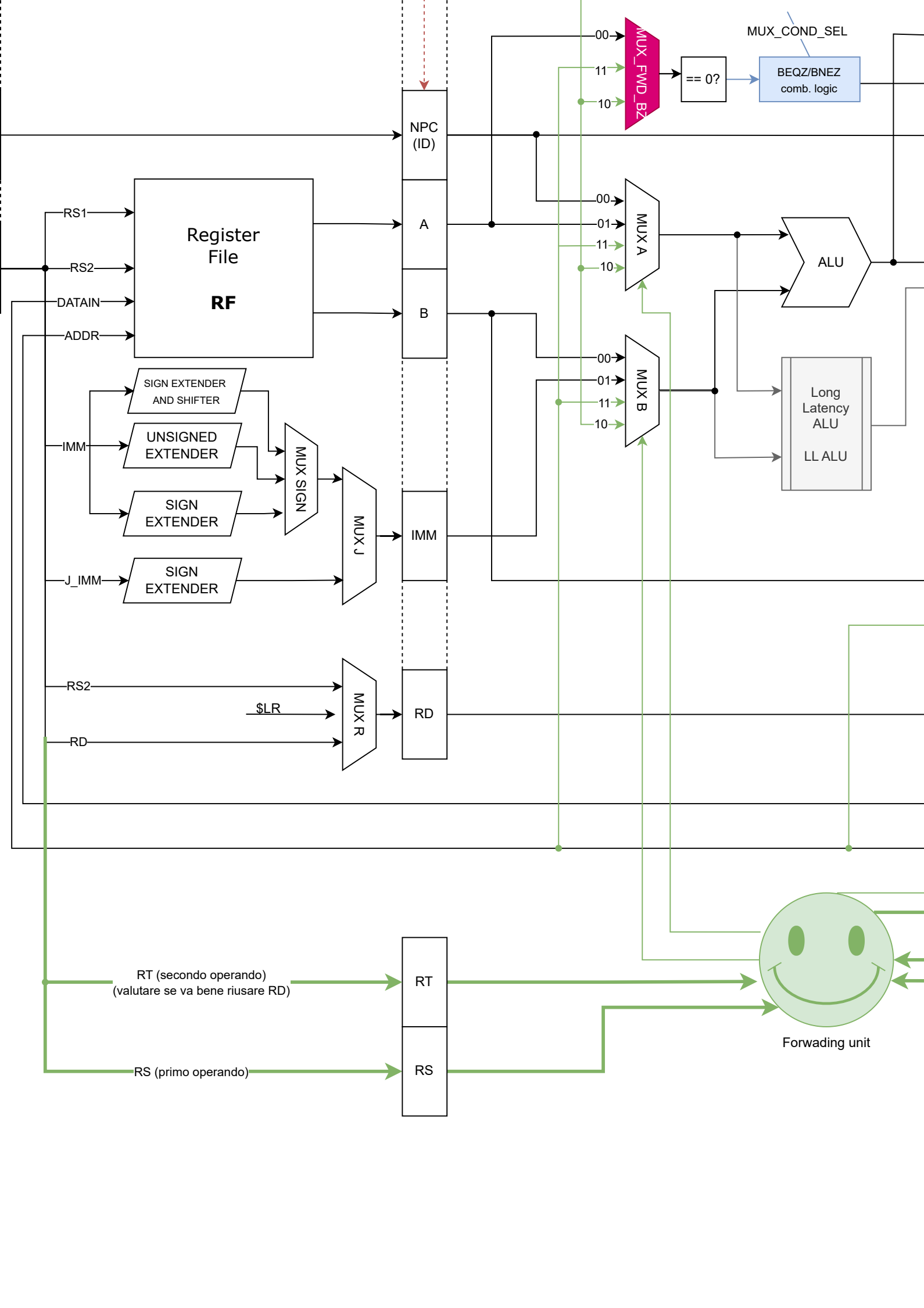
In più devo mettere una nop dopo la fase di exe

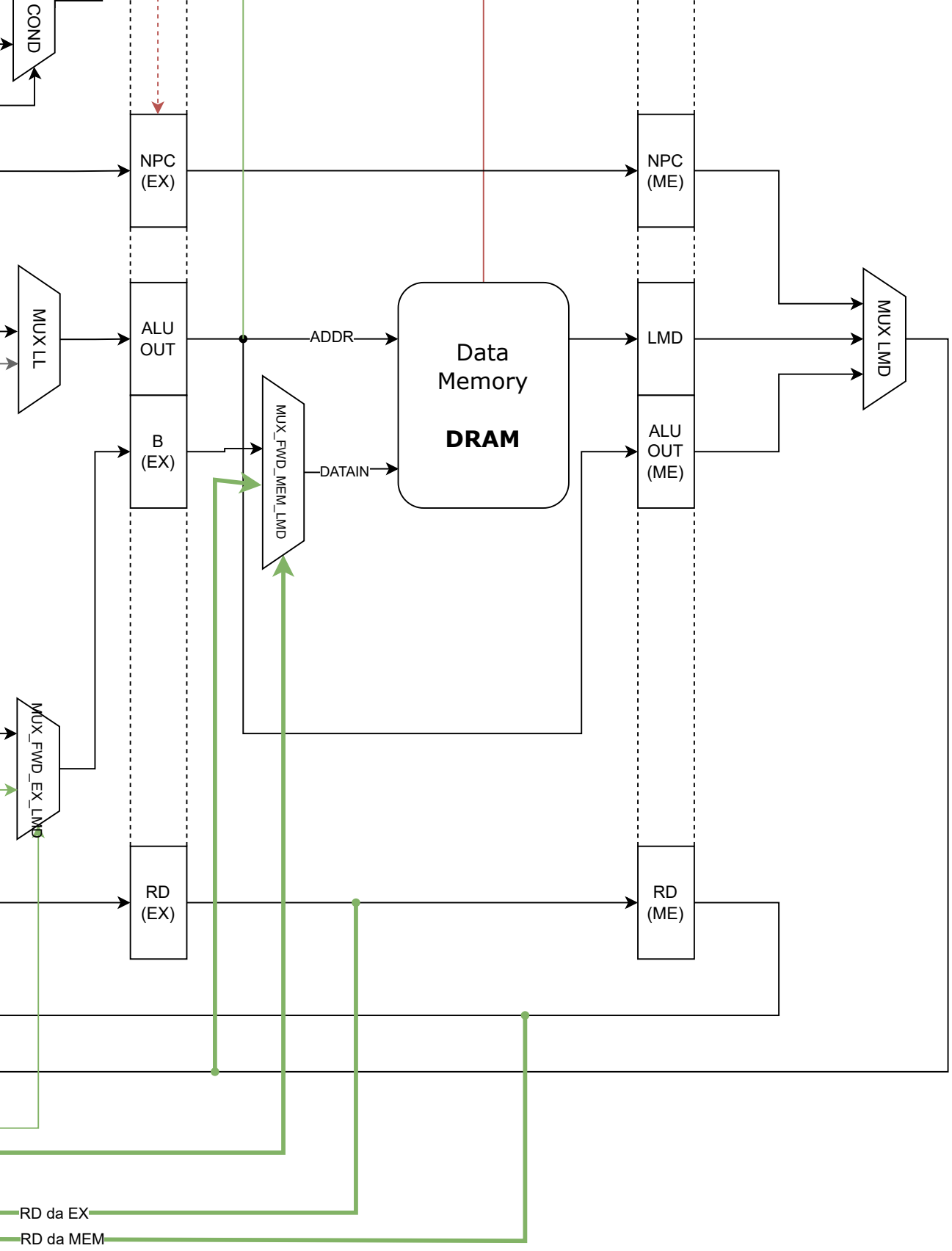
EX_EN

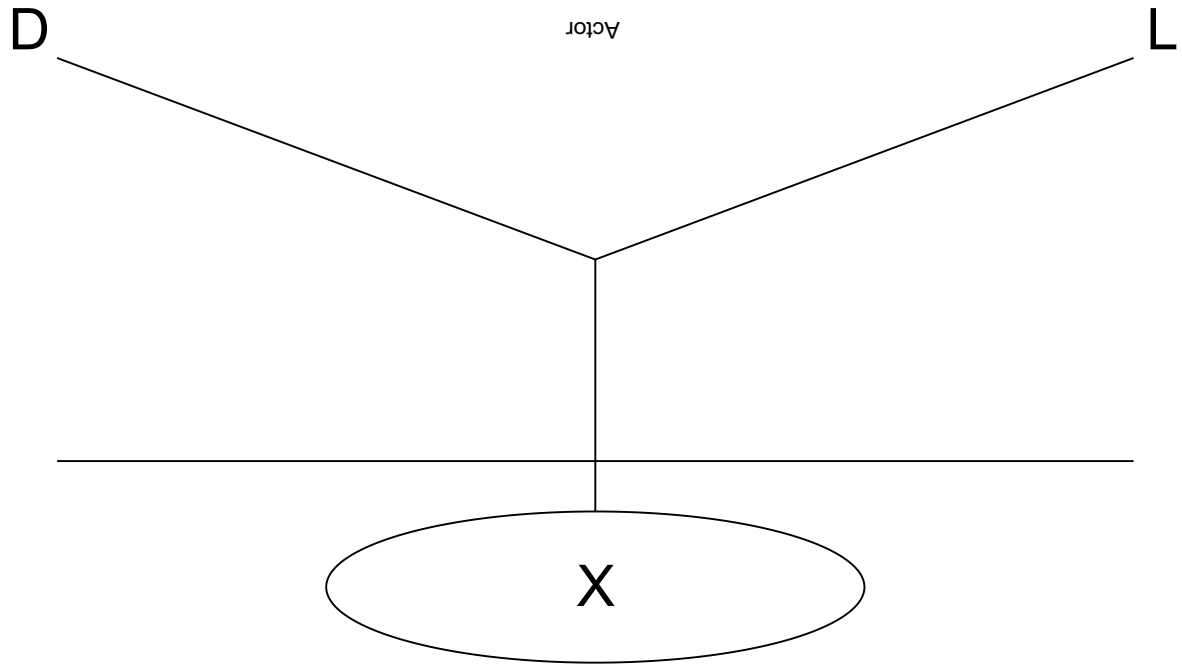



NOP_CW_MUX
CW3
NOP_CW_MUX
CW4
MUX


1
+
NPC
PC
MUXJUMP
Instruction
Memory
IRAM
NOP_MUX
NOP
IR
non c'è questo in violetto,
permetterebbe di ridurre
lo stallo della jump di 1


MUX_COND_SEL
00
11
10
MUX_FWD_BZ
== 0?
BEQZ/BNEZ
comb. logic
NPC
(ID)
RS1
RS2
DATAIN
ADDR
Register
File
RF
A
B
00
01
11
10
MUX A
ALU
00
01
11
10
MUX B
Long
Latency
ALU
LL ALU
SIGN EXTENDER
AND SHIFTER
IMM
UNSIGNED
EXTENDER
SIGN
EXTENDER
MUX SIGN
MUX J
IMM
J_IMM
SIGN
EXTENDER
RS2
$LR
RD
MUX R
RD
RT (secondo operando)
(valutare se va bene riusare RD)
RT
RS (primo operando)
RS
Forwading unit


COND
NPC
(EX)
NPC
(ME)
MUX LL
ALU
OUT
ADDR
Data
Memory
DRAM
LMD
MUX LMD
B
(EX)
MUX_FWD_MEM_LMD
DATAIN
ALU
OUT
(ME)
MUX_FWD_EX_LMD
RD
(EX)
RD
(ME)
RD da EX
RD da MEM



Stallare in decode :

F D E M W

|-------

F D - - - -> fetch come istruzione dopo la decode diventano nop

F D E M W -> add

il registro SCRITTO nella fase di decode deve essere scritto normalmente

il registro LETTO nella fase di decode deve essere disabilitato -> vero?

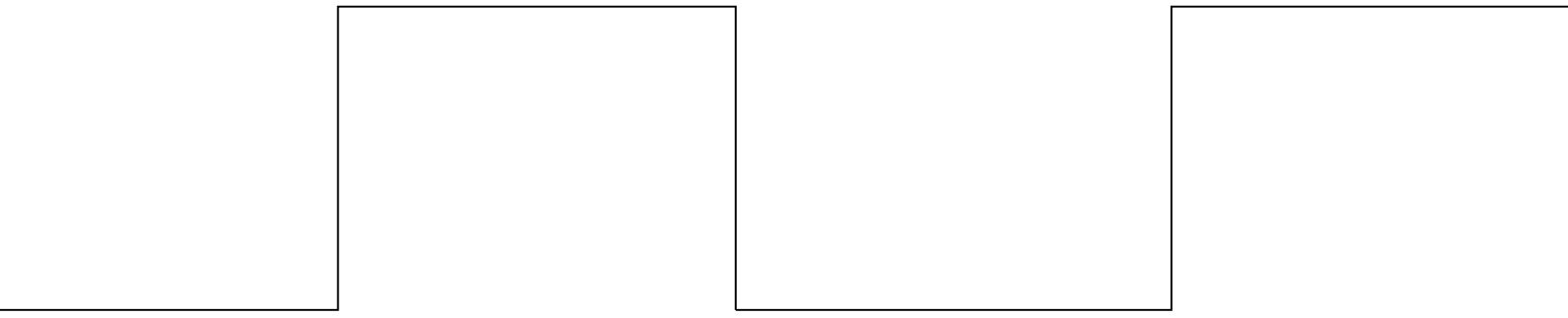
E' vero perché (?) dobbiamo mantenere in IF/ID, i dati della istruzione cl
stalla in decode.

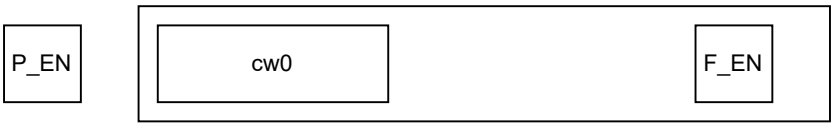
Quindi

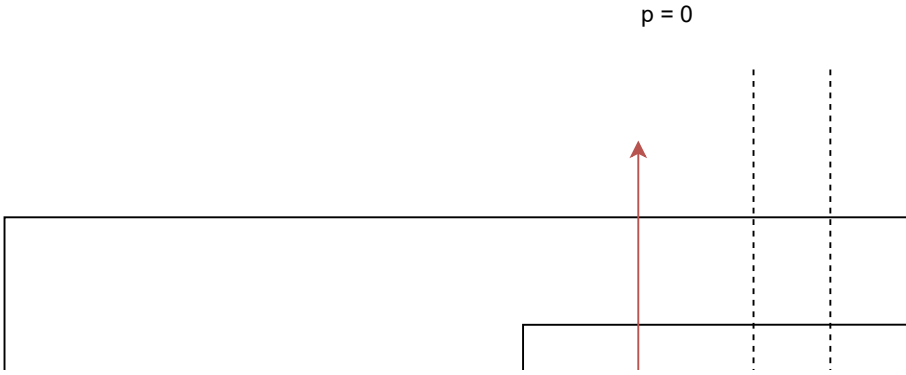
Questo avviene durante il rising edge della fase, ovvero disabilito i registri
sono scritti al falling

ne

che



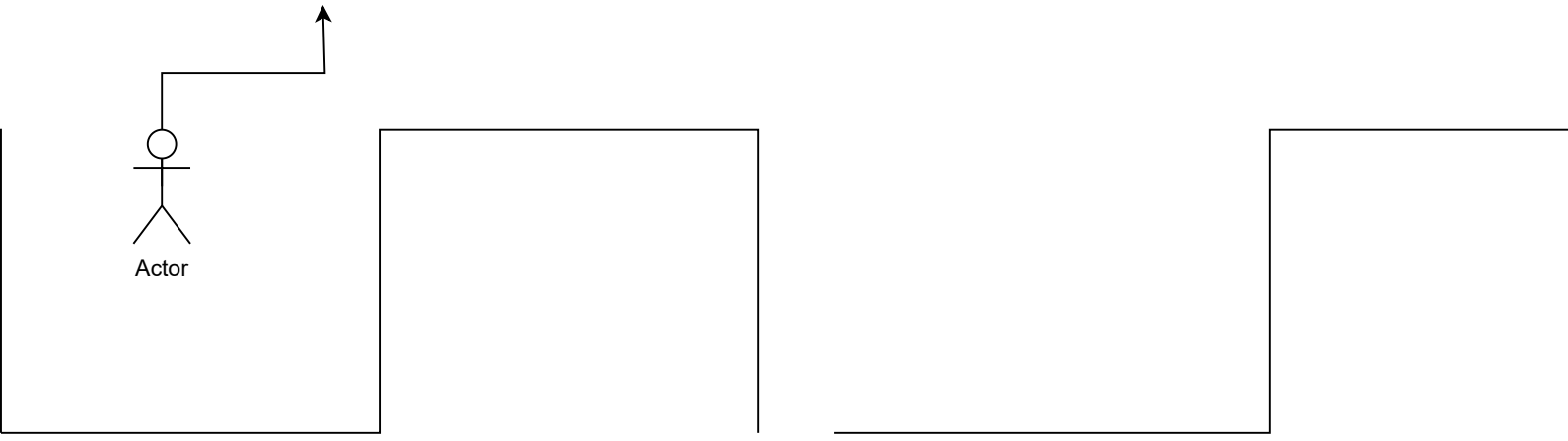

P_EN
cw0
F_EN

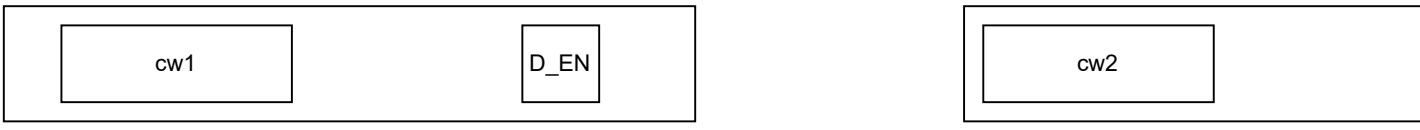

p = 0

se c'è uno stallo in decode
  enable.fetch = false   # disabilito i registri scritti in fase di fetch

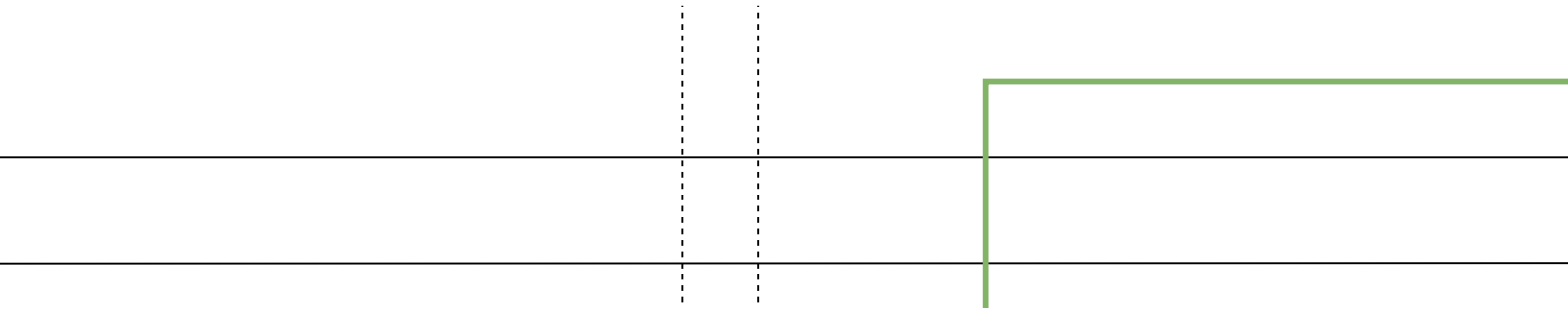
a cw da mandare dopo la decode deve essere una nop
  e.g. add 'stalla in decode' significa che non può andare in execute
  quindi la cw in execute deve essere quella di una nop.
  La cw non modifica lo stato ma devo ritardare l'arrivo della cw add in ex
  mantenendo il valore di cw1 (control word della ins in fase di decode)
  uguale a quella della add. Quindi IN TEORIA,

  if enable.fetch
    cw1 <= cw0   # Aggiornato con la cw della instruzione in fetch
  else
    cw1 unchanged
    cw2 <= NOP

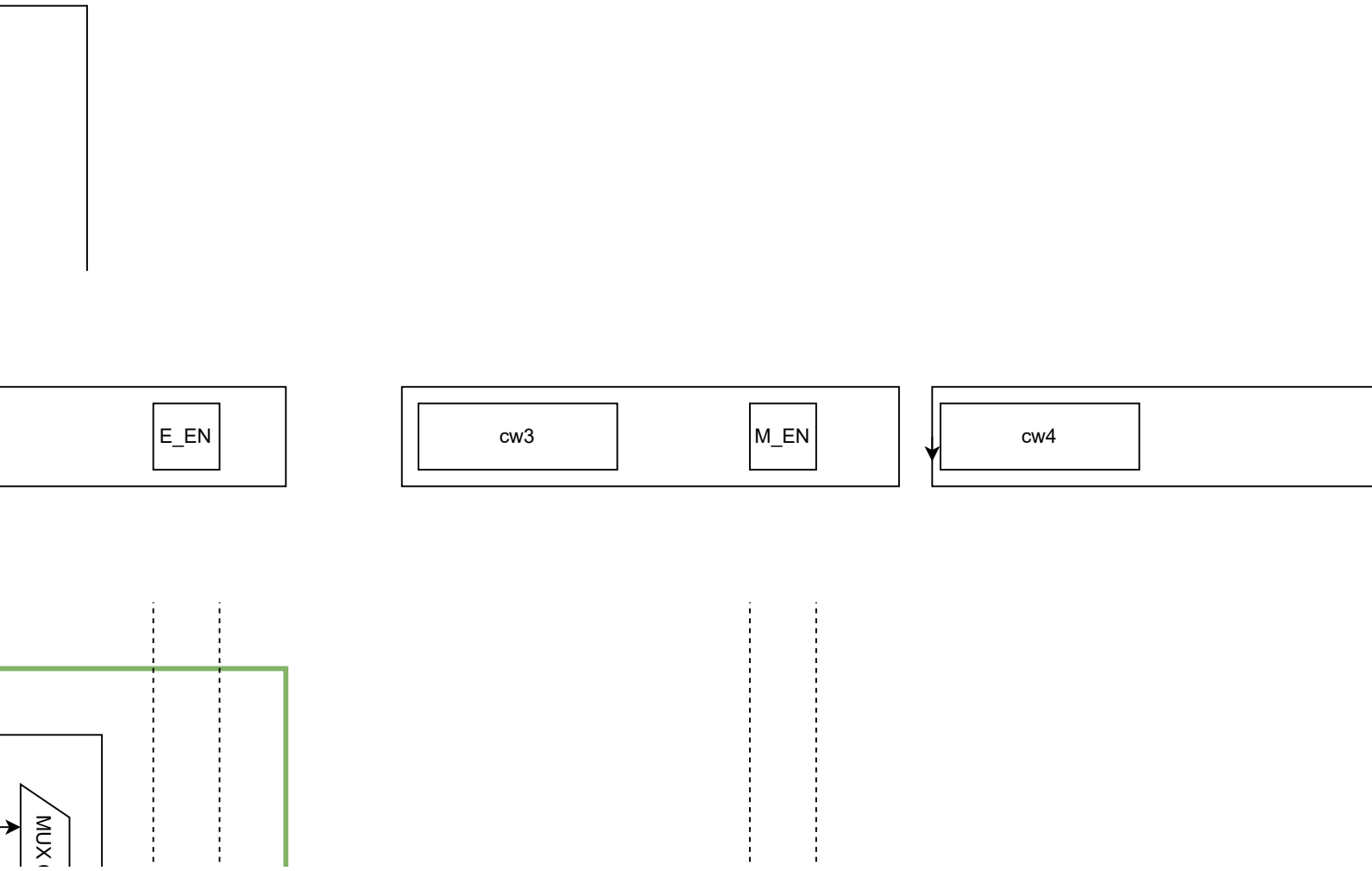


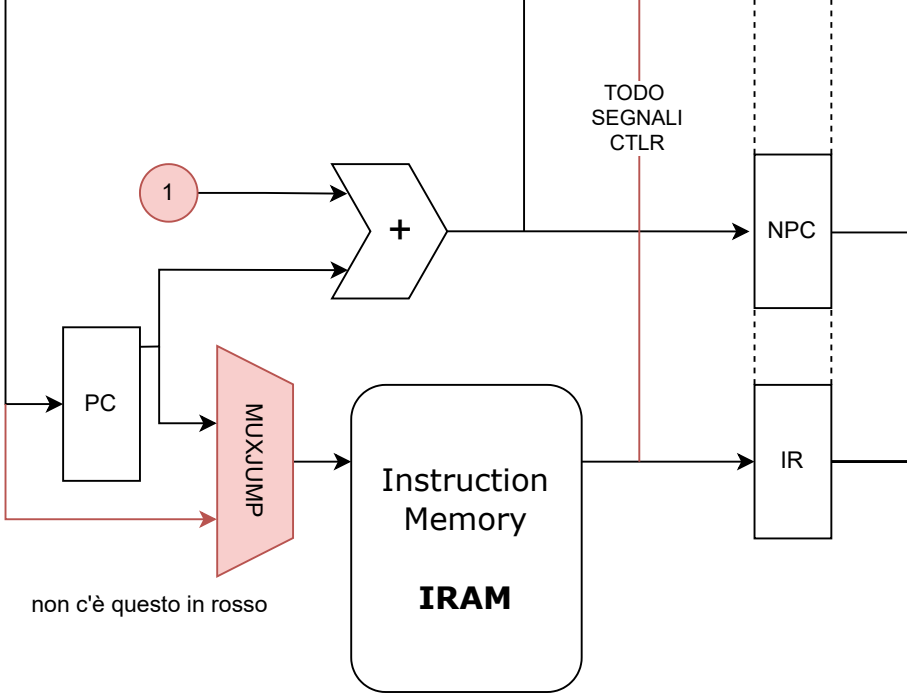


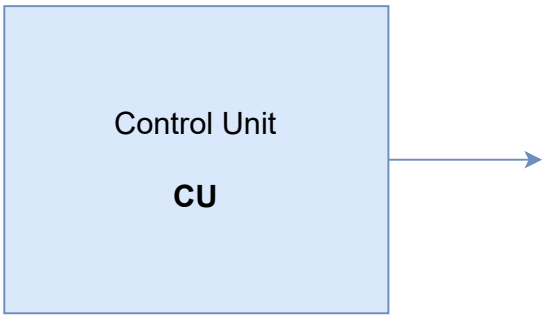
aggiorna cw1 con la jump p = 1

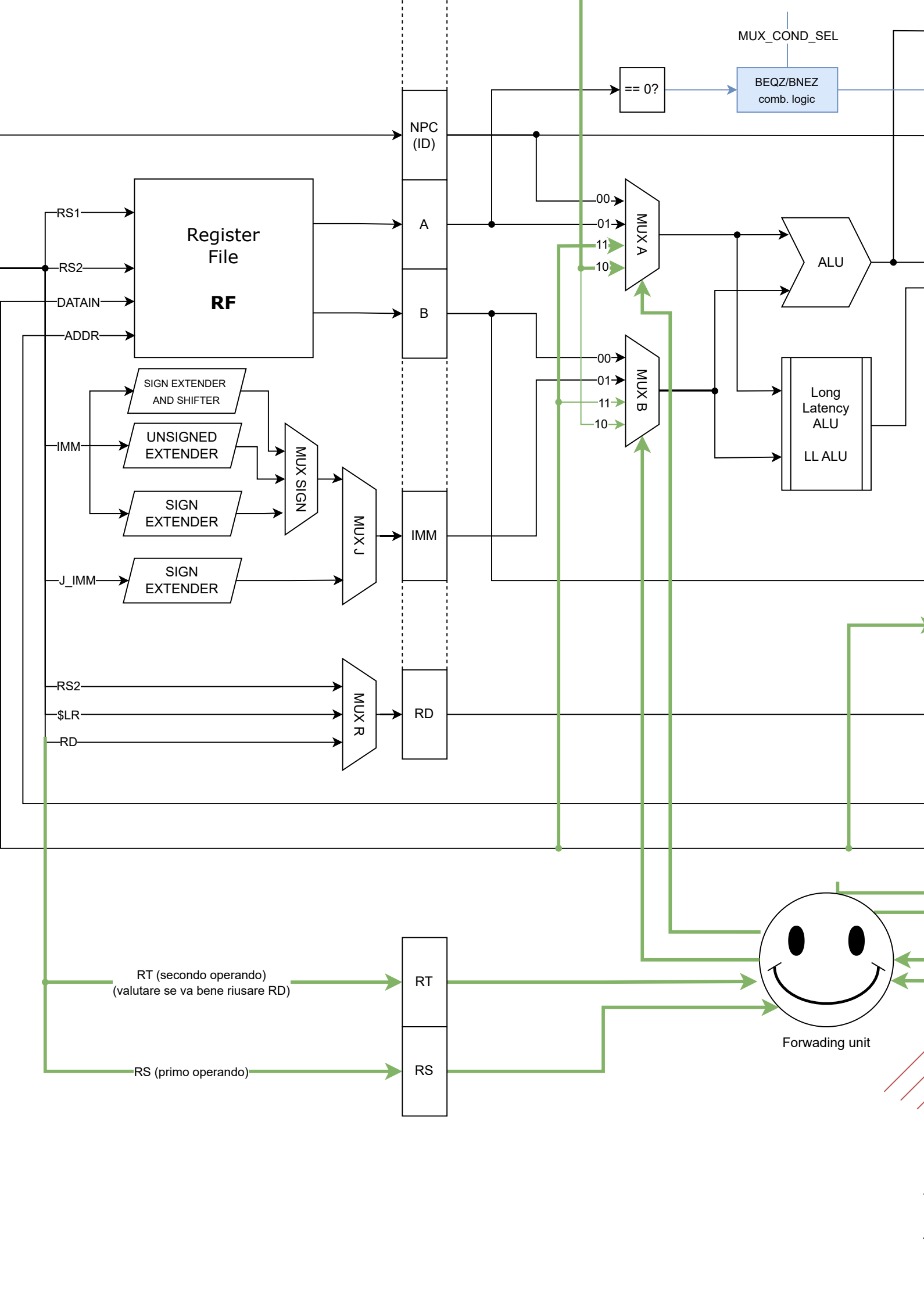


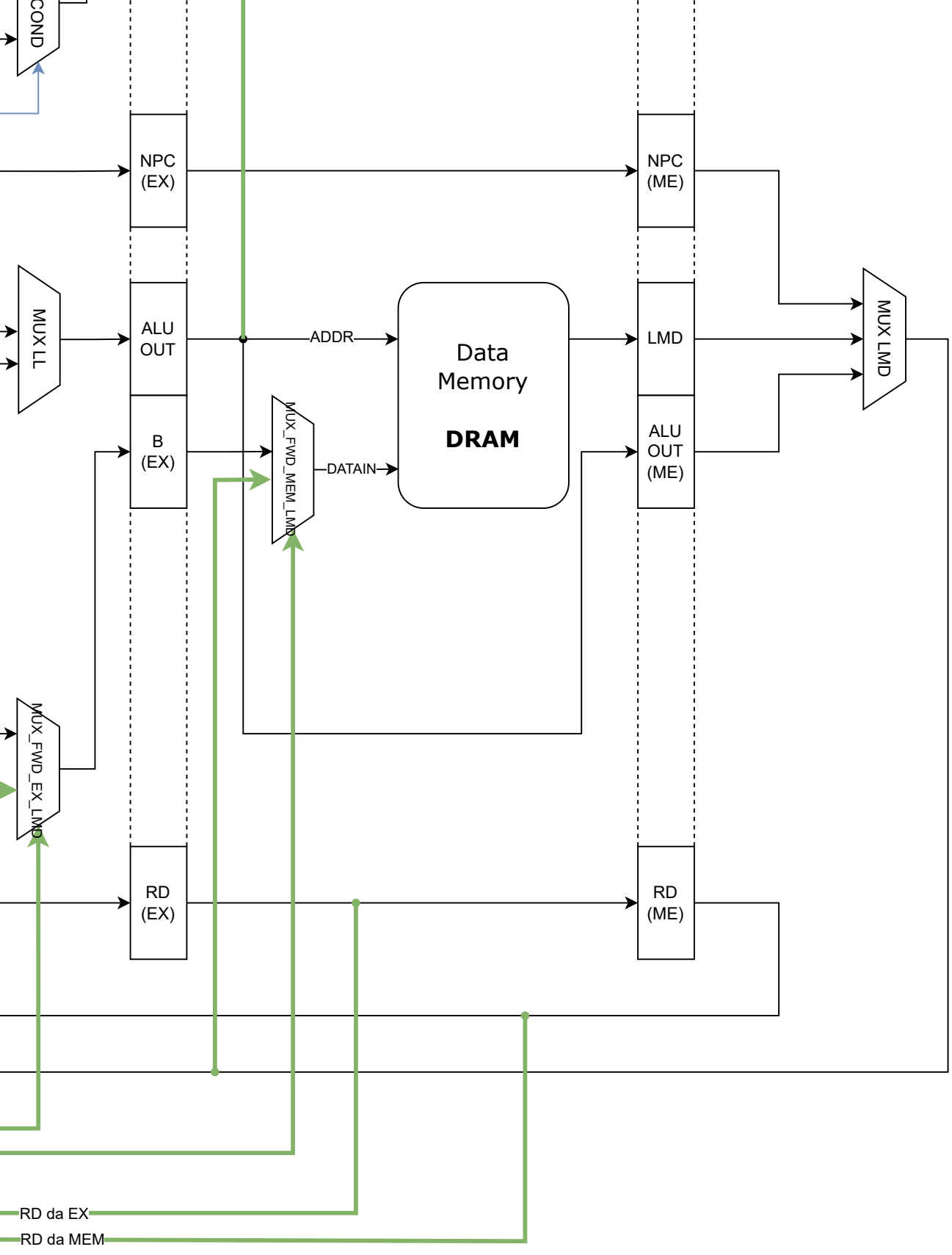
e,

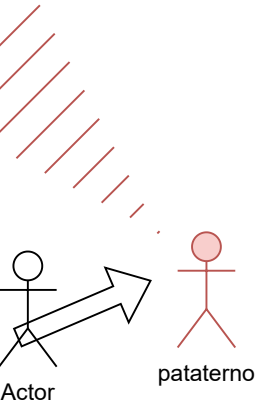


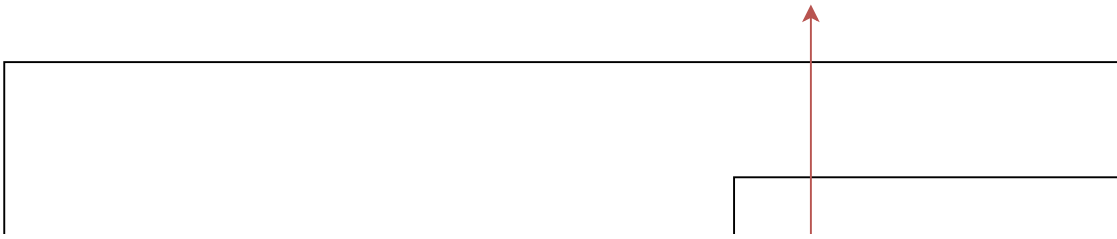

TODO
SEGNALI
CTLR
1
+
NPC
PC
MUXJUMP
Instruction
Memory
IRAM
IR
non c'è questo in rosso

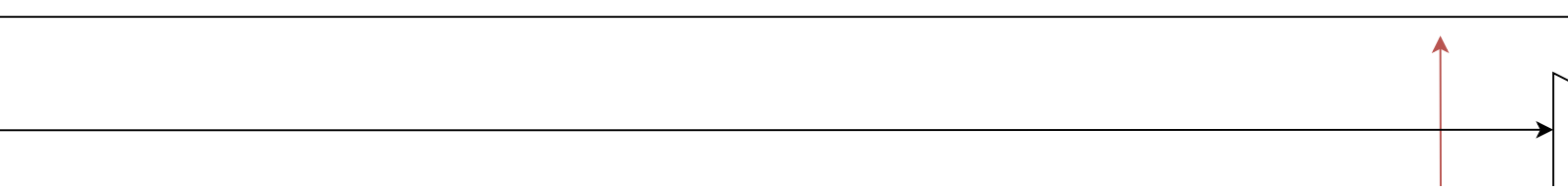

Control Unit
CU


MUX_COND_SEL
== 0?
BEQZ/BNEZ
comb. logic
NPC
(ID)
A
B
IMM
RD
RT
RS
Register
File
RF
RS1
RS2
DATAIN
ADDR
00
01
11
10
MUX A
MUX B
ALU
Long
Latency
ALU
LL ALU
SIGN EXTENDER
AND SHIFTER
UNSIGNED
EXTENDER
SIGN
EXTENDER
MUX SIGN
MUX J
IMM
J_IMM
SIGN
EXTENDER
RS2
$LR
RD
MUX R
RT (secondo operando)
(valutare se va bene riusare RD)
RS (primo operando)
Forwading unit

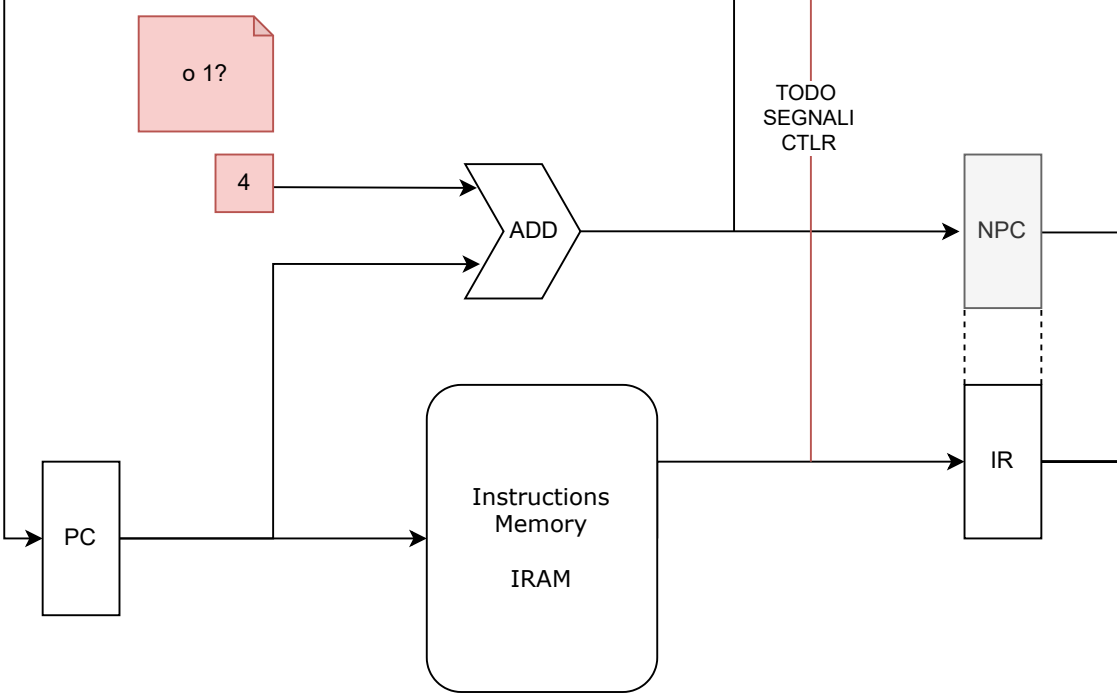

COND
NPC (EX)
NPC (ME)
MUX LL
ALU OUT
ADDR
Data Memory
DRAM
LMD
MUX LMD
B (EX)
MUX_FWD_MEM_LMD
DATAIN
ALU OUT (ME)
MUX_FWD_EX_LMD
RD (EX)
RD (ME)
RD da EX
RD da MEM

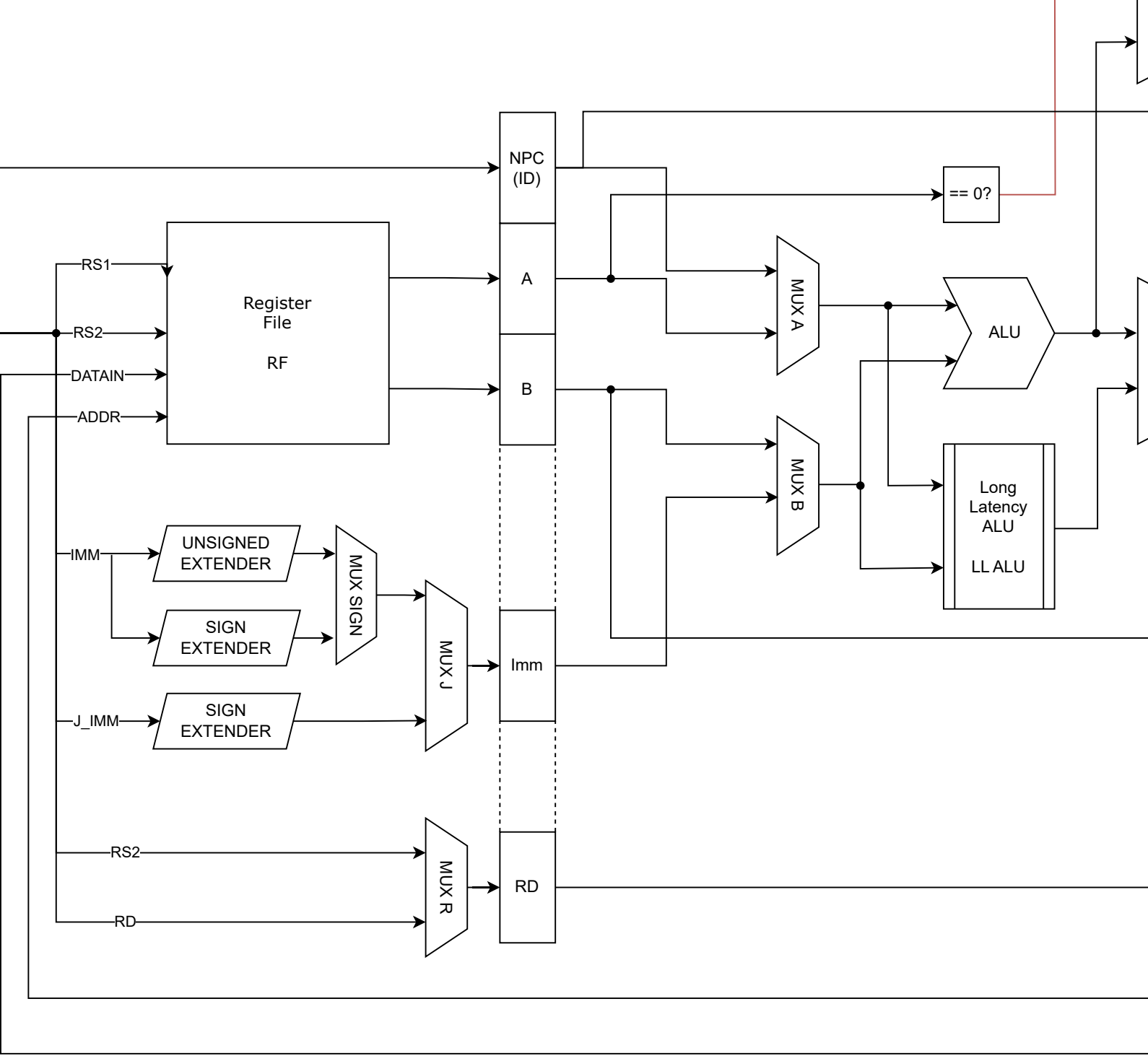

Actor
pataterno

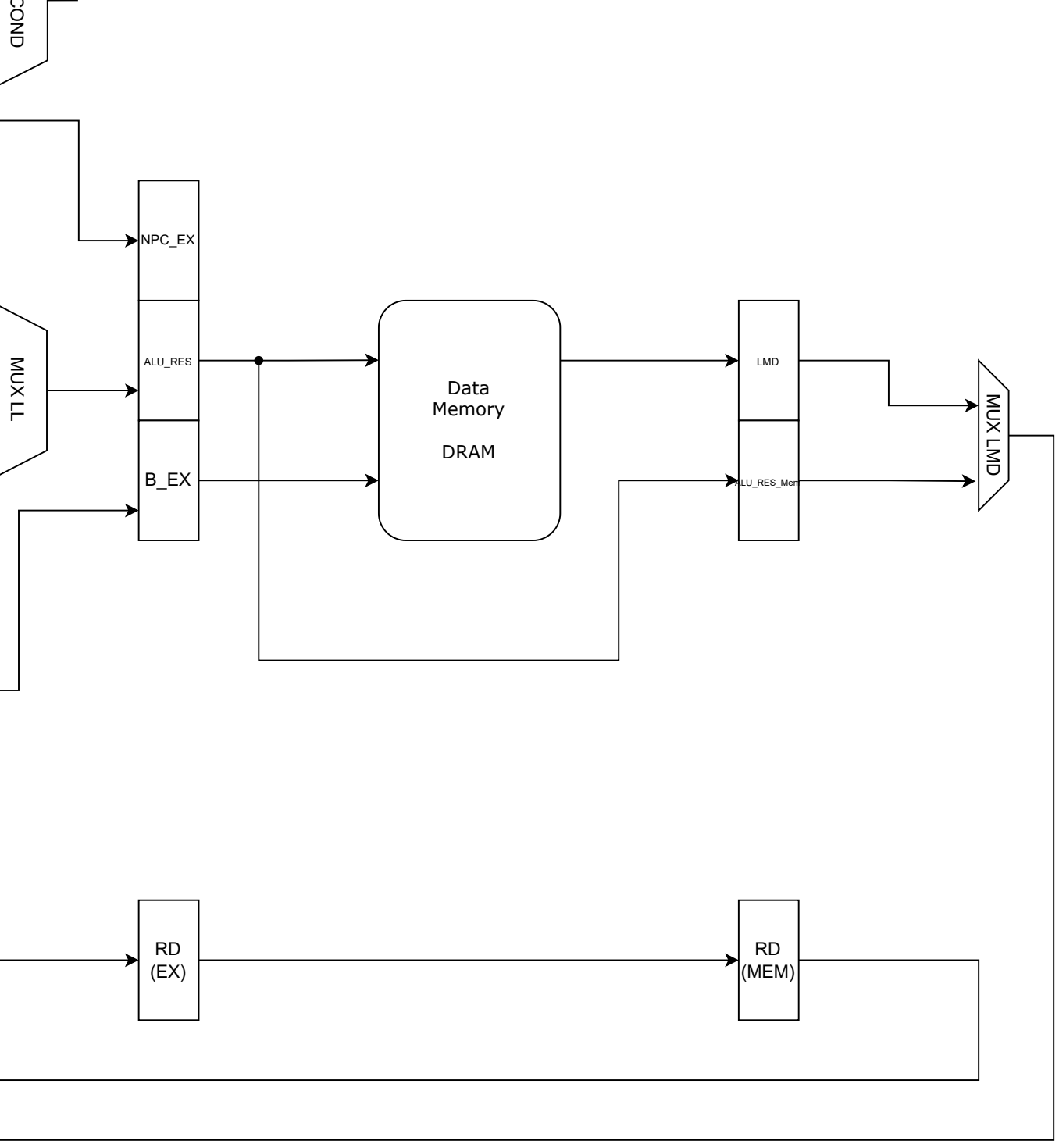




MUX


o 1?
4
ADD
TODO
SEGNALI
CTLR
NPC
IR
PC
Instructions
Memory
IRAM


NPC
(ID)
== 0?
RS1
RS2
DATAIN
ADDR
Register
File
RF
A
B
MUX A
ALU
MUX B
Long
Latency
ALU
LL ALU
IMM
UNSIGNED
EXTENDER
SIGN
EXTENDER
MUX SIGN
MUX J
Imm
J_IMM
SIGN
EXTENDER
RS2
RD
MUX R
RD


COND
NPC_EX
MUX LL
ALU_RES
B_EX
Data
Memory
DRAM
LMD
ALU_RES_Mem
MUX LMD
RD
(EX)
RD
(MEM)