



DRAM\_RDY

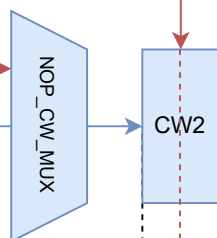
SEGNALE DIRETTAMENTE  
DA IRAM, O passa dalla cu?

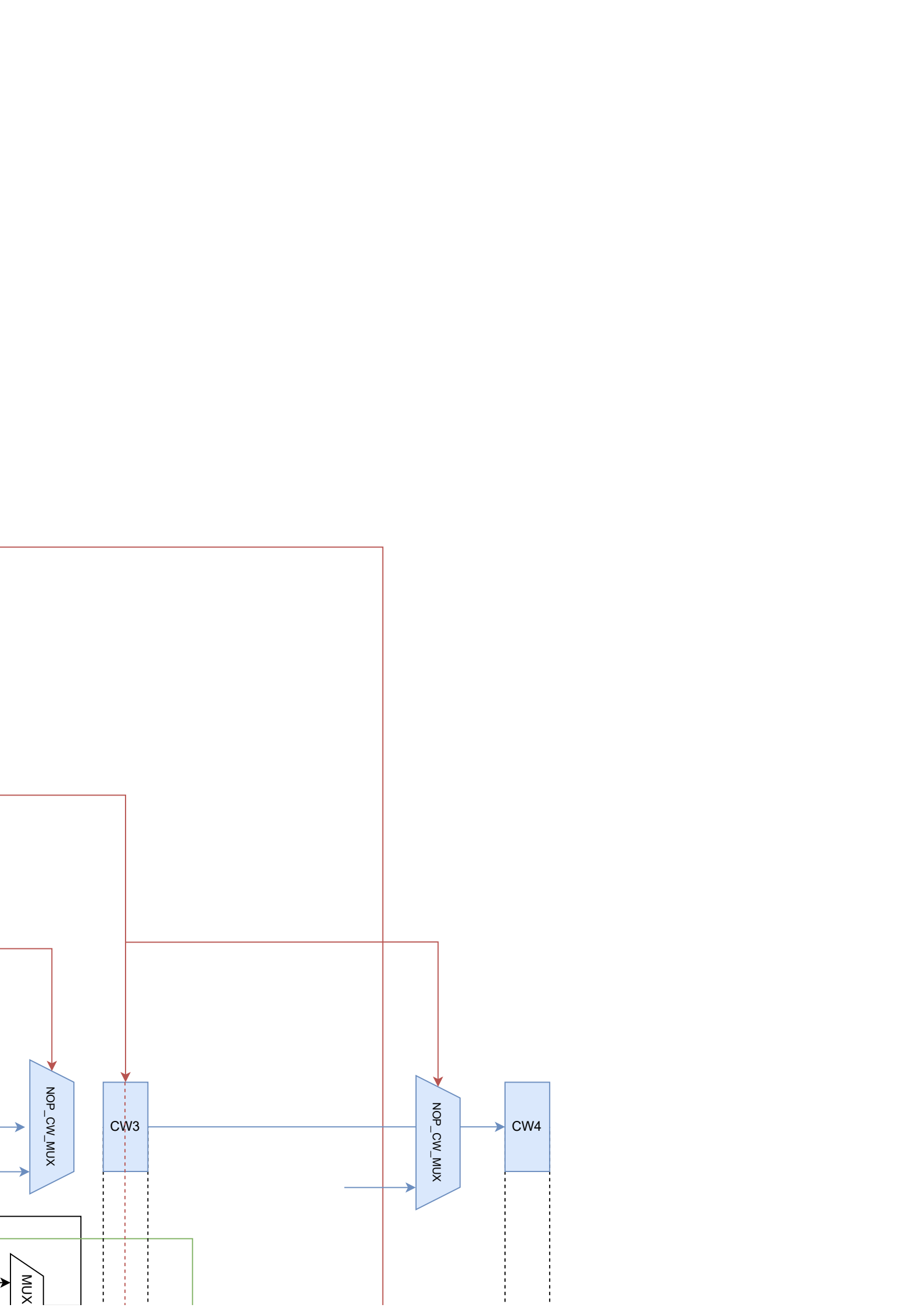
devo disabilitare == prevenire che vengano aggiornati  
pc  
if/id  
id/exe  
exe/mem

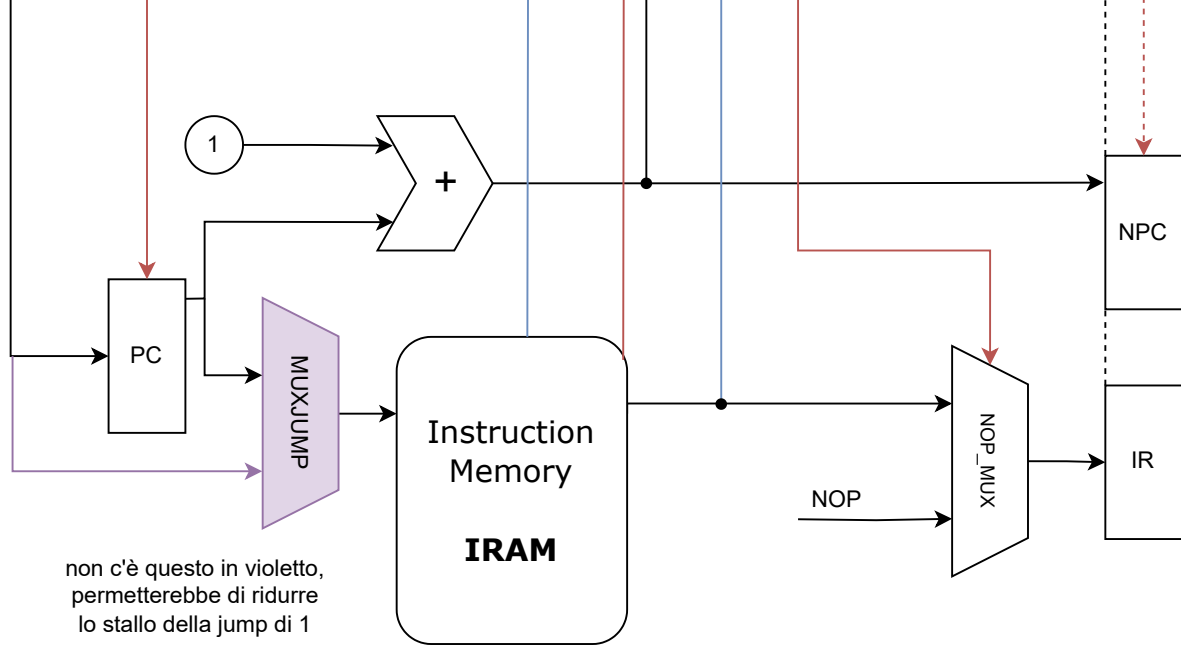
In più devo mettere una nop dopo la fase di exe

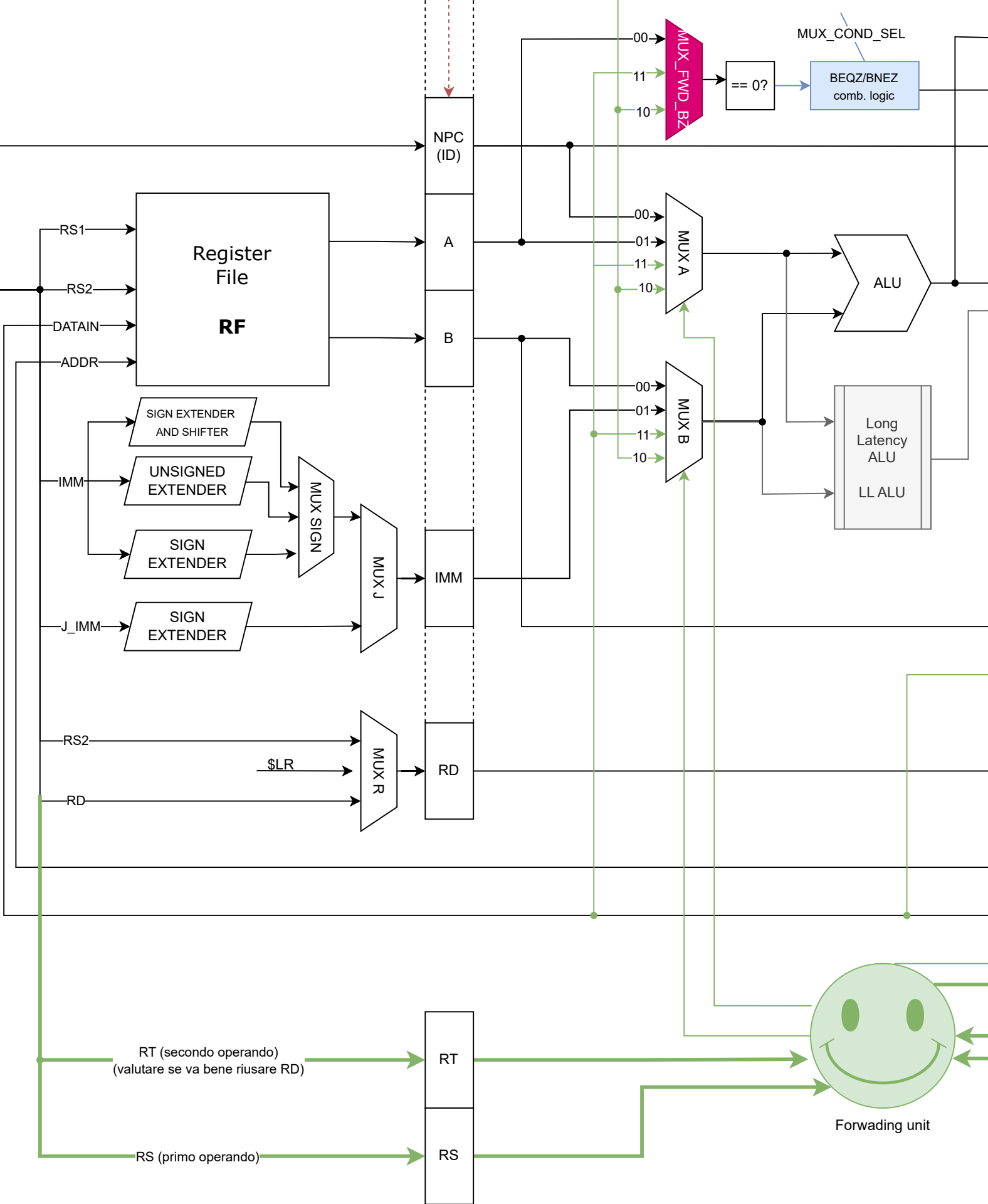
EX\_EN

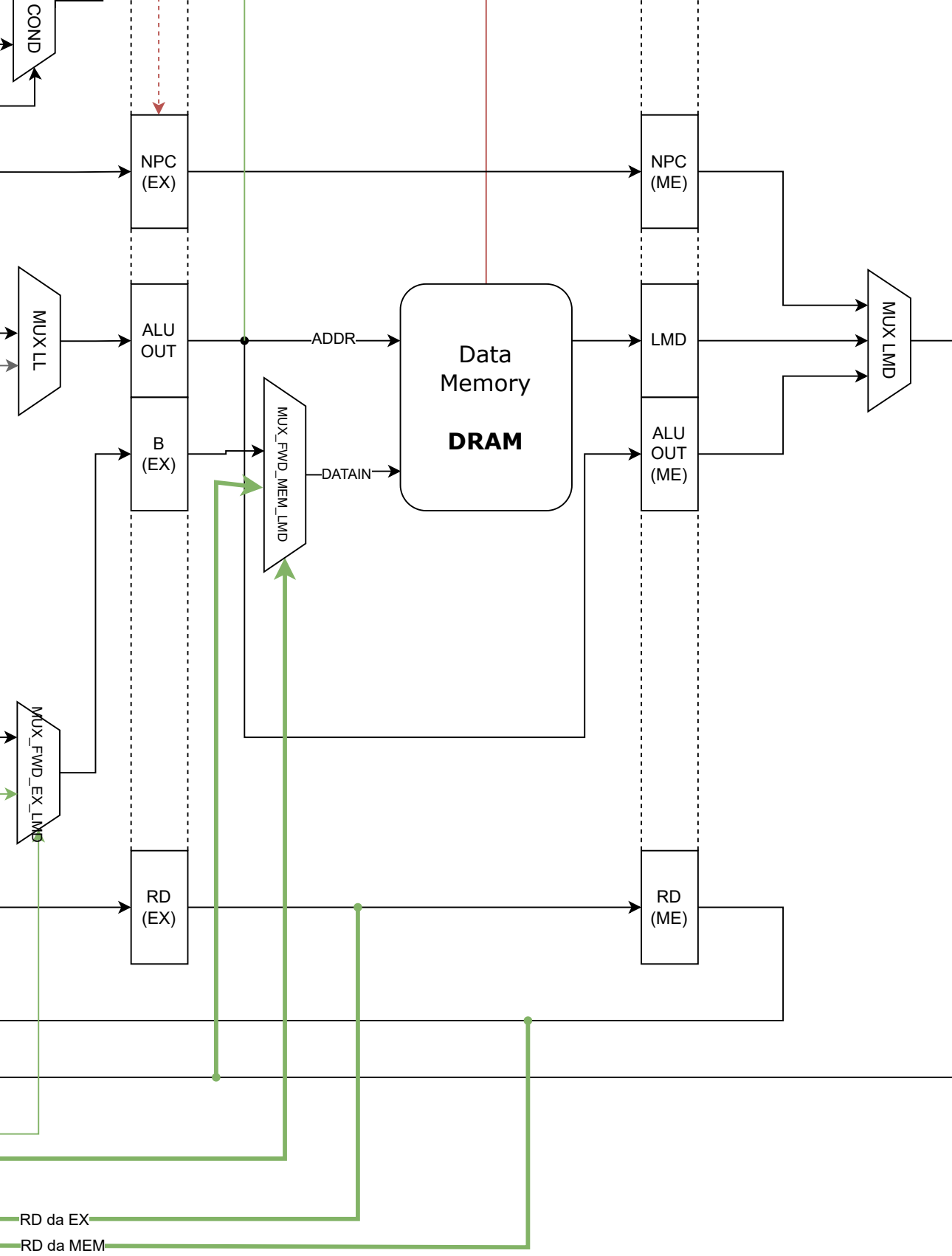
ID\_EN





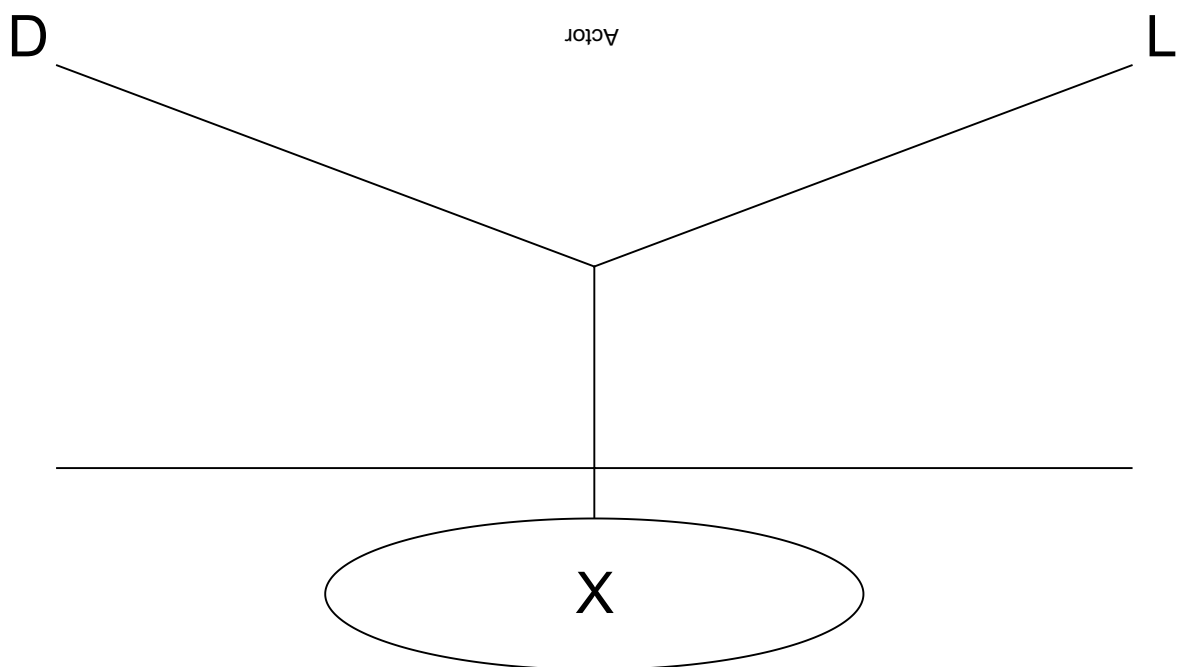












Stallare in decode :

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 che stalla in decode.

Quindi

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

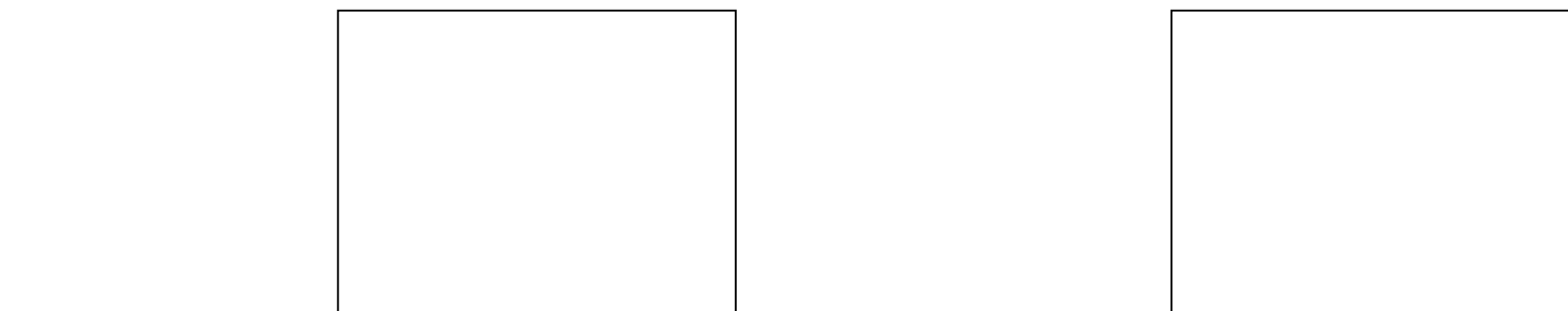


ne

che





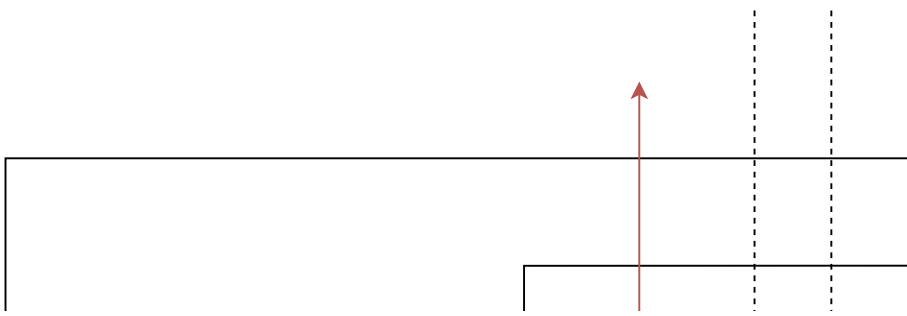


P\_EN

cw0

F\_EN

$p = 0$



se c'è uno stallo in decode

enable.fetch = false      # disabilita i registri scritti in fase di fetch

a cw da mandare dopo la decode deve essere una nop

e.g. add 'stallo 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,

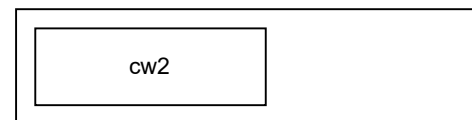
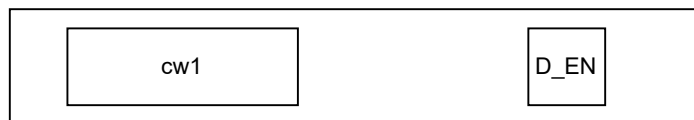
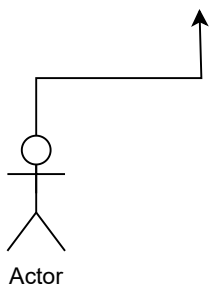
if enable.fetch

    cw1 <= cw0      # Aggiornato con la cw della istruzione in fetch

else

    cw1 unchanged

    cw2 <= NOP



aggiorna cw1 con la jump

p = 1



e,

