

CARP MEETING 8

Control Transfer and
Branch Prediction



cal-poly-ramp.github.io

AUTHOR: RYAN CRAMER



BRANCH- CEPTION

MUSIC COMPOSED BY HANS ZIMMER

Branch vs Jump?

BRANCHES



JUMPS



High Level:

What does { } mean in C?

SCOPE: declares a stack

Maybe:

- Initialize variables
- Receive function arguments
- Operate on function arguments
- Return stuff

High Level:

Why do we need branch/jump

- Branches:
 - Enables “if/else”
 - Enables “for”, “while”
- Jumps:
 - Enables function calls
 - Enables call, return

WHAT DO WE NEED TO JUMP/BRANCH?

[if, else, while,
for, do-while,
return, ternary]



US

shutterstock.com · 127357478

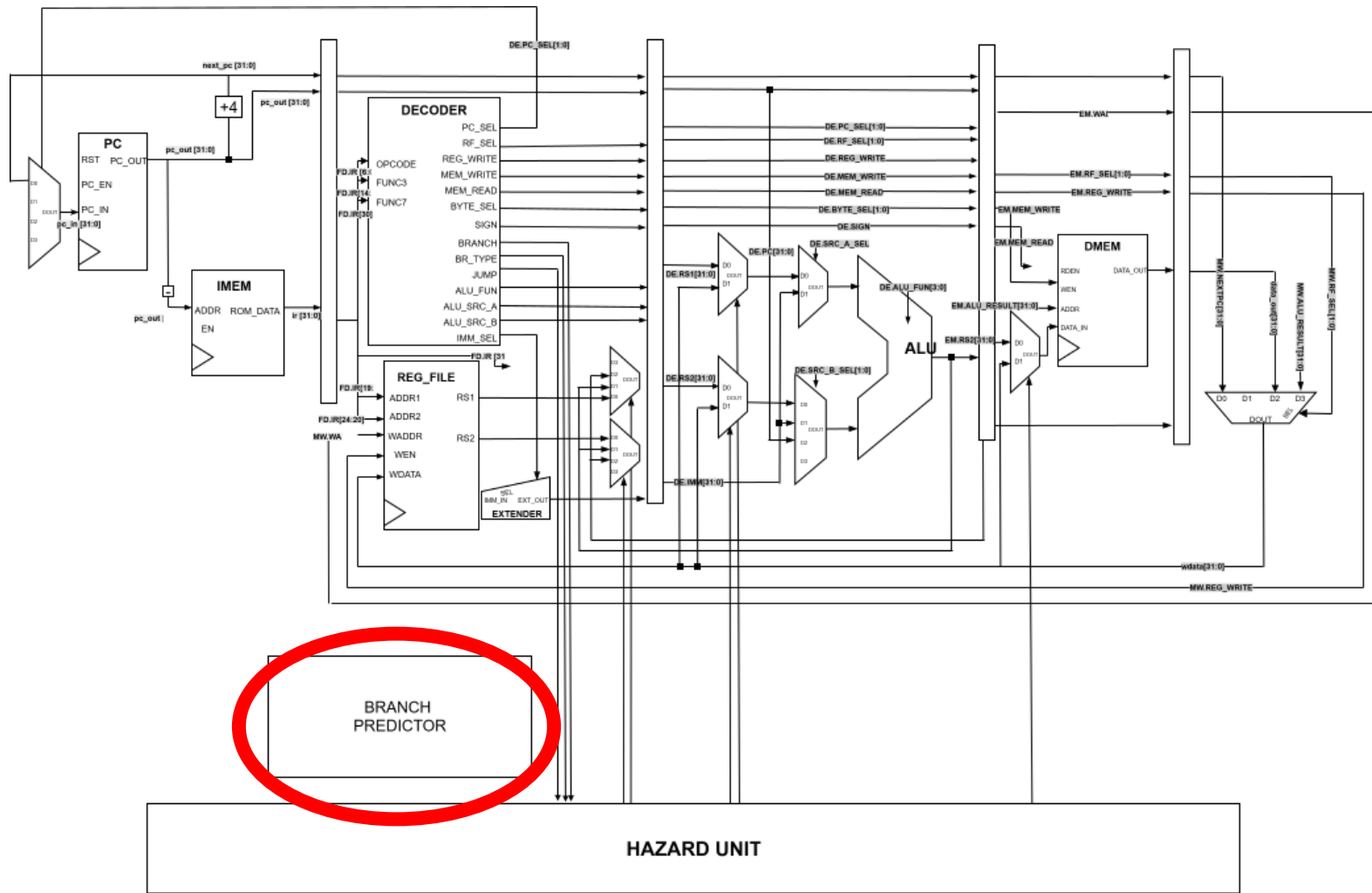
To branch/jump, we need:

Three OPCODES:

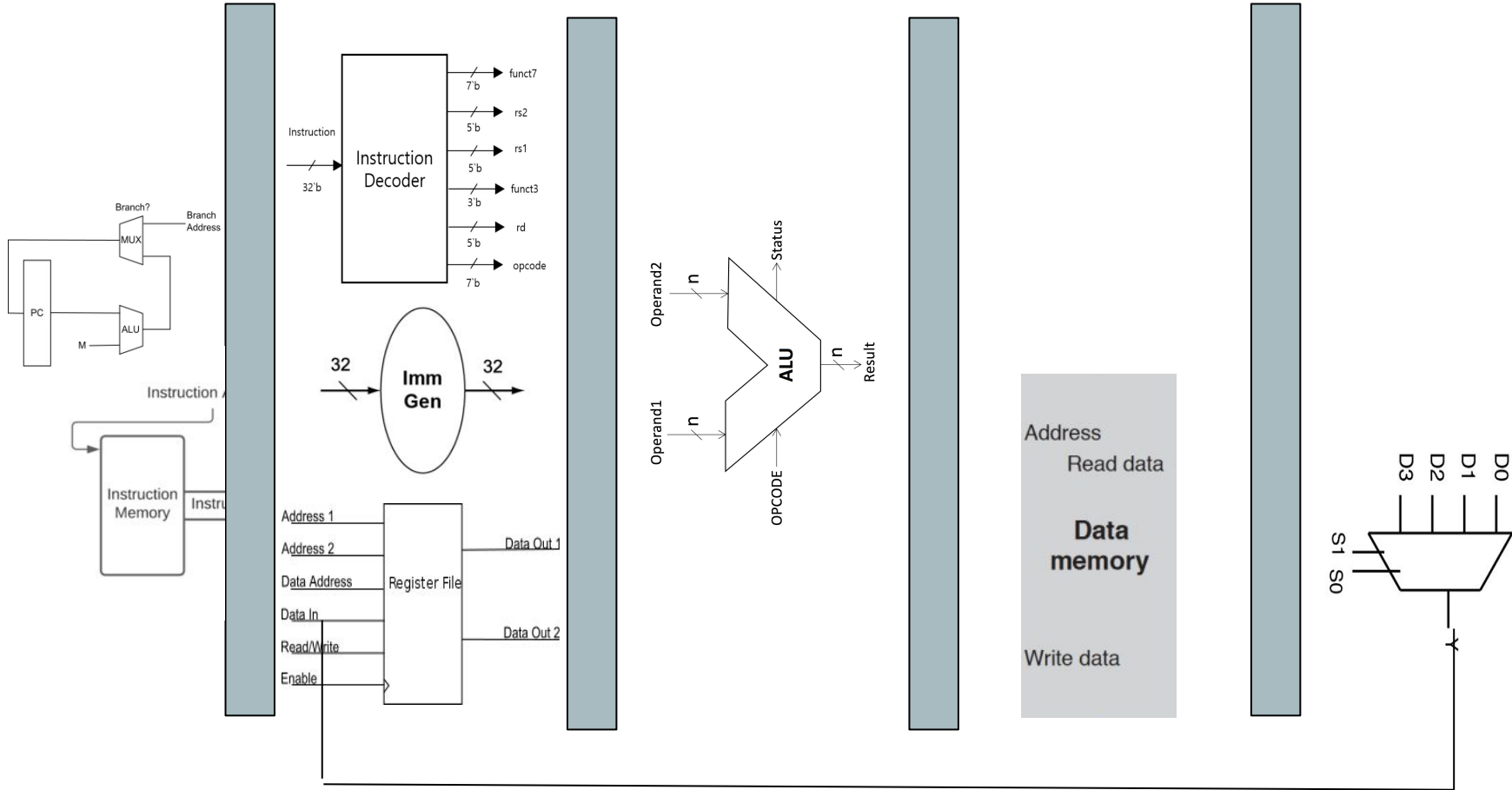
JAL: need **PC + J_Type**

JALR: need **PC + (RS1 + I_Type)**

BRANCH: need **PC + B_type + Comparator(RS1, RS2)**

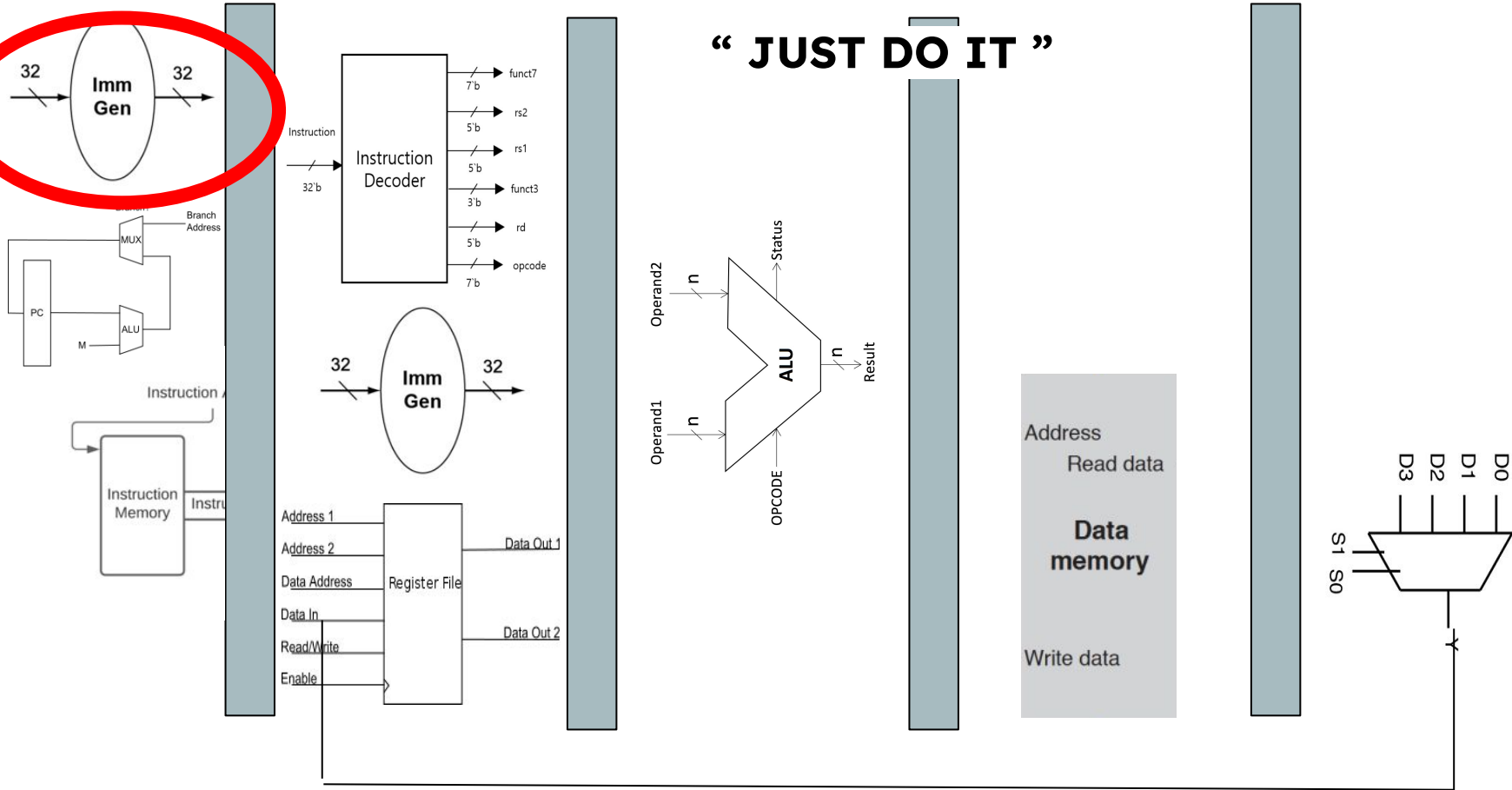


JAL, JALR, BRANCH

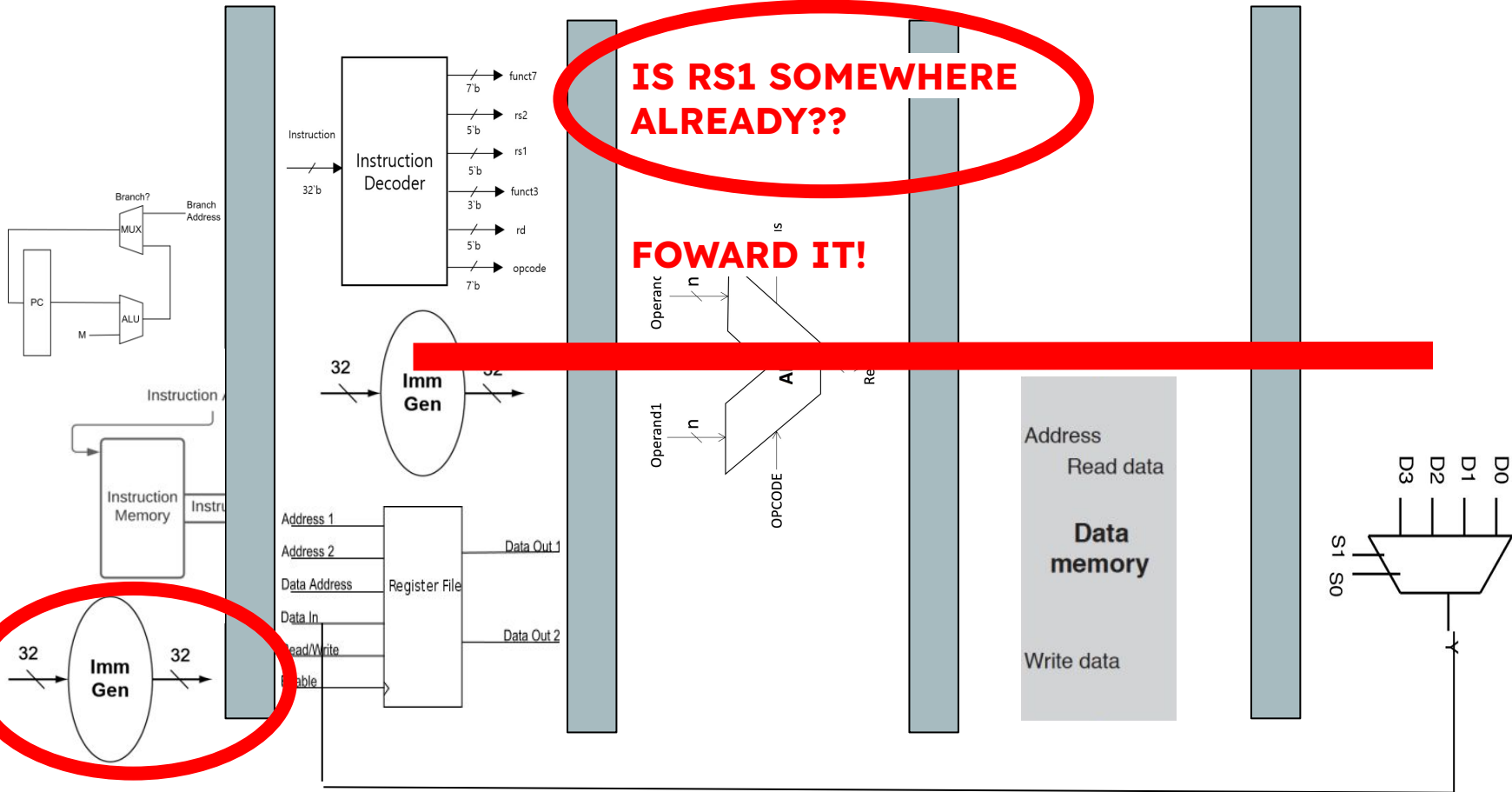


JAL FETCH HANDLING

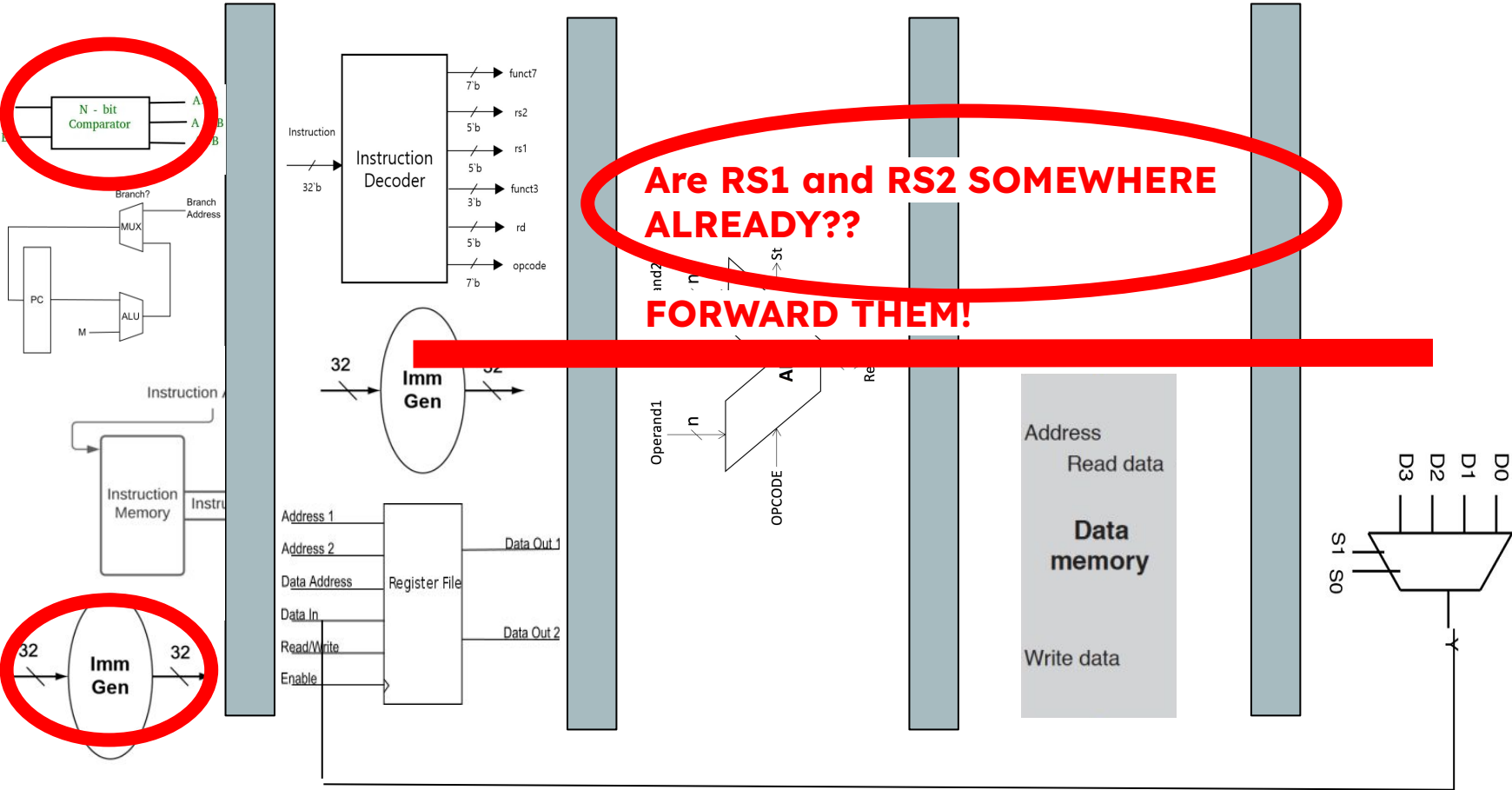
“ JUST DO IT ”



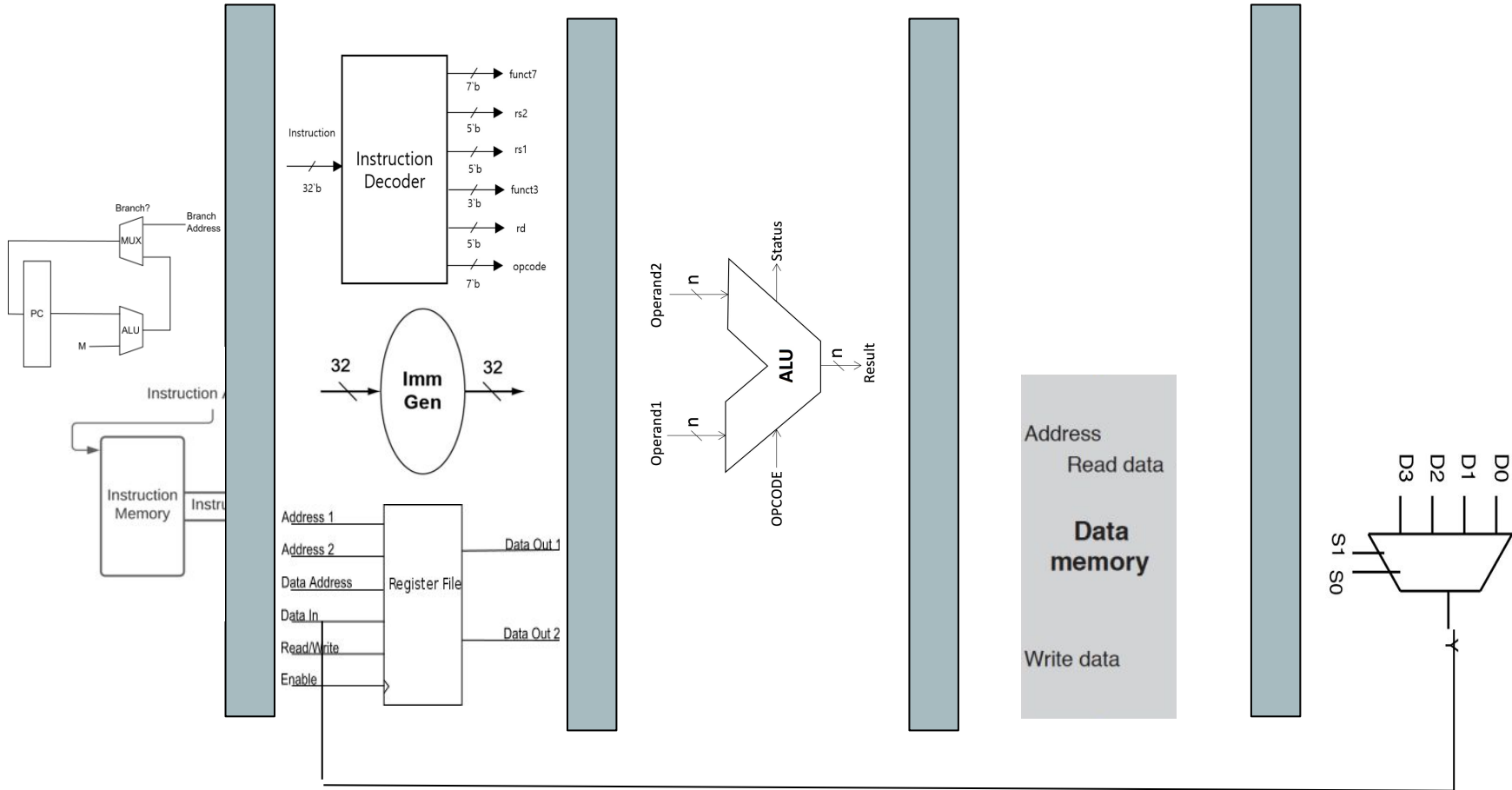
JALR FETCH HANDLING



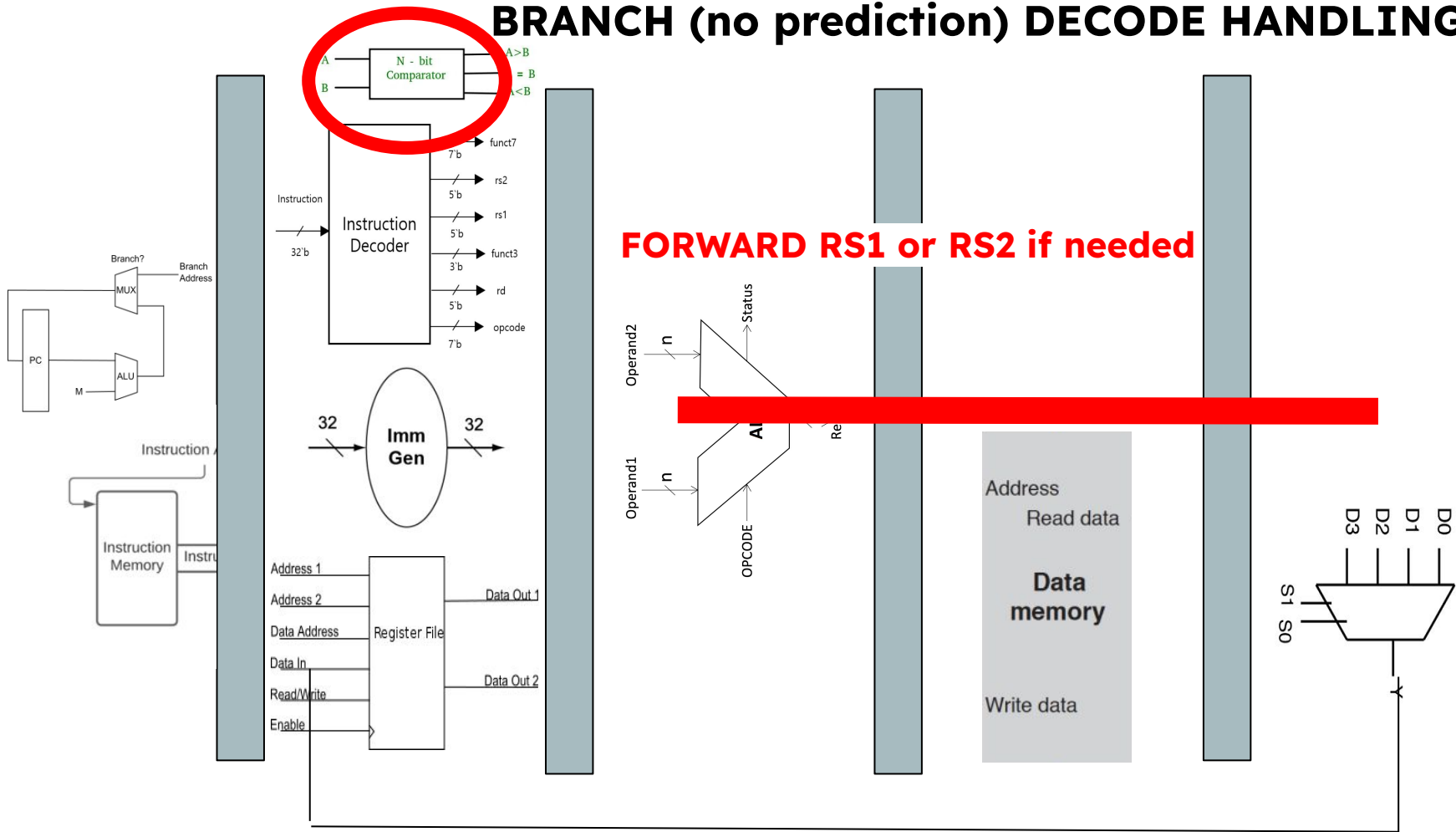
BRANCH (no prediction) FETCH HANDLING



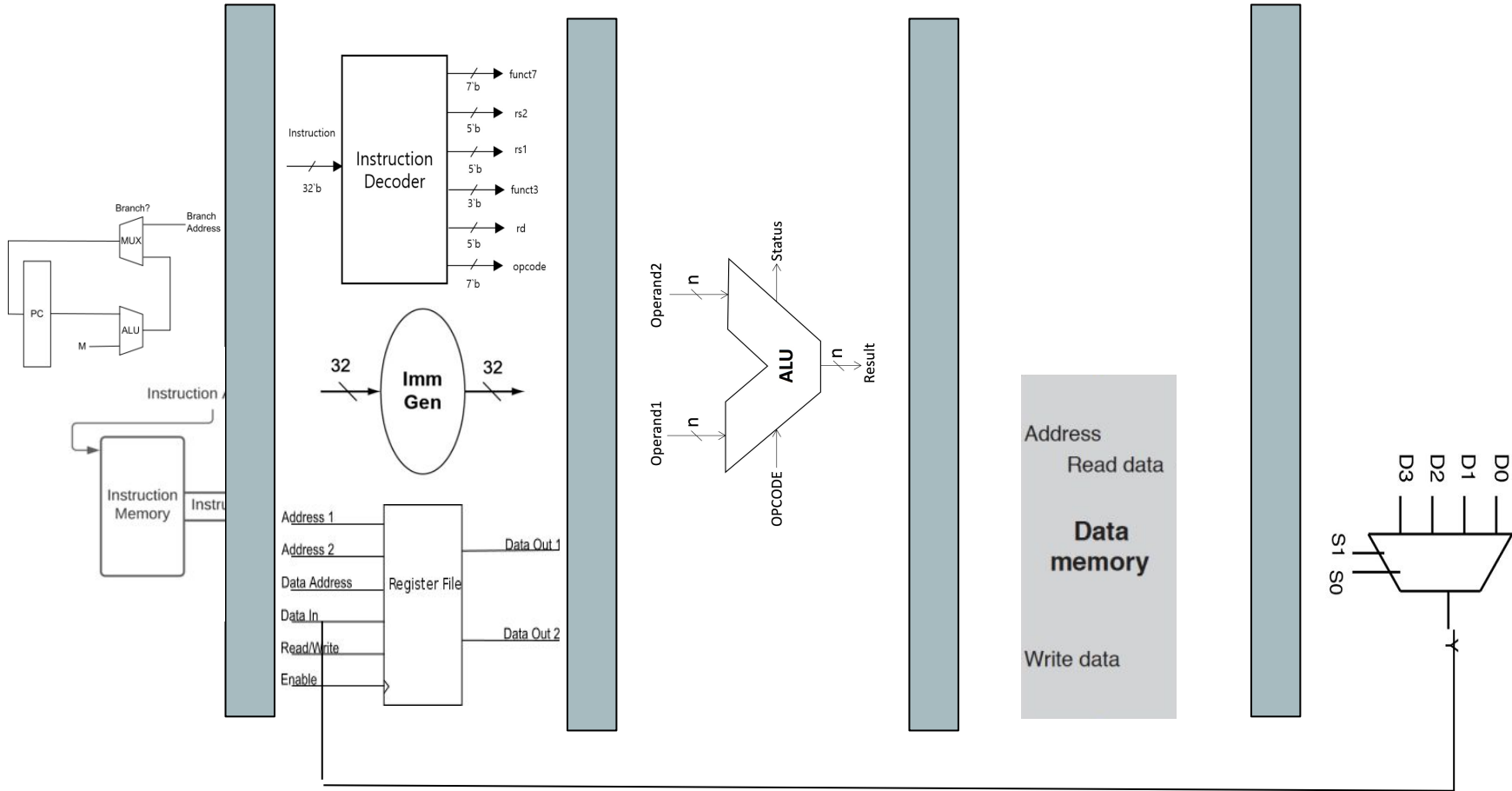
JALR DECODE HANDLING



BRANCH (no prediction) DECODE HANDLING



No Prediction BRANCH Execute Handling



We forgot to do hazards



Hazards:

Jal:

- An unresolved branch/jump is ahead

Jalr:

- An unresolved branch/jump is ahead
- RS1 is in multiply-hazard
- Divider stall is active
- Load-use for RS1

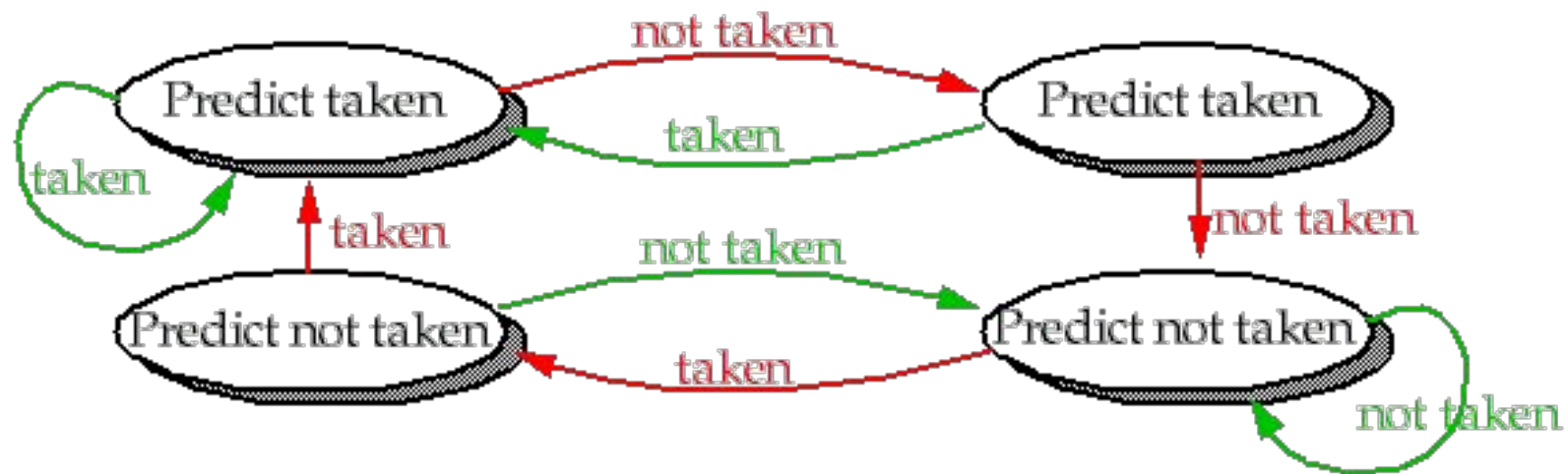
Branch:

- Obviously RS1, RS2 hazards
 - Unresolved branch/jump ahead
-

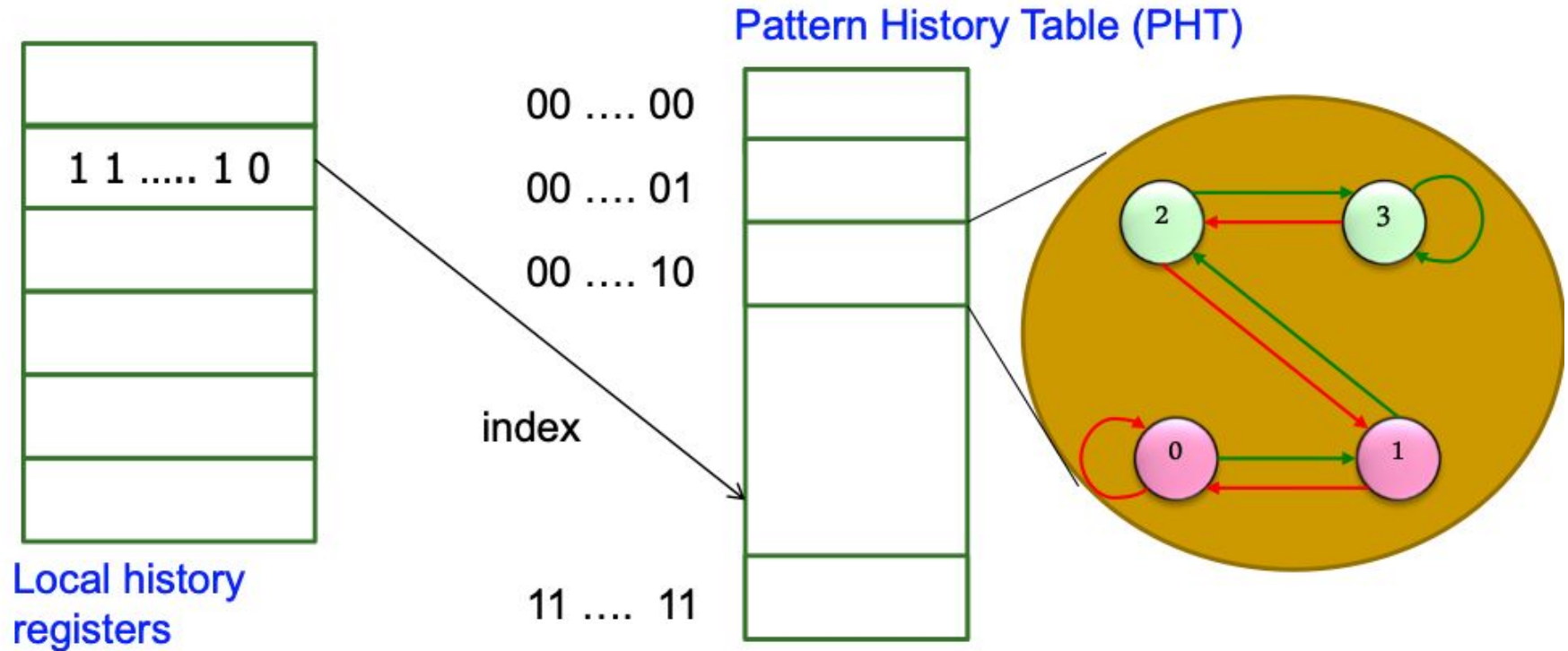
We finally made it....



Branch Prediction



EARLIEST FORM OF MACHINE LEARNING!



H: HISTORY BIT

P: PREDICTION

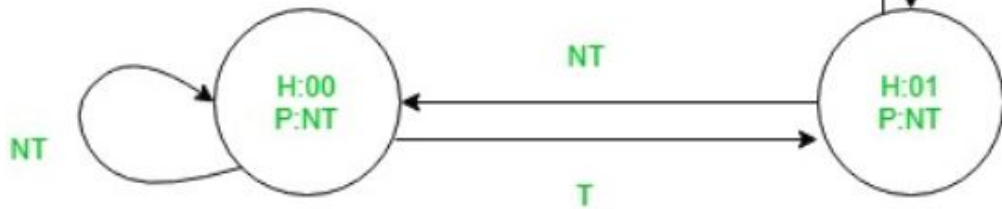
STRONGLY
TAKEN

WEAKLY
TAKEN



T

NT



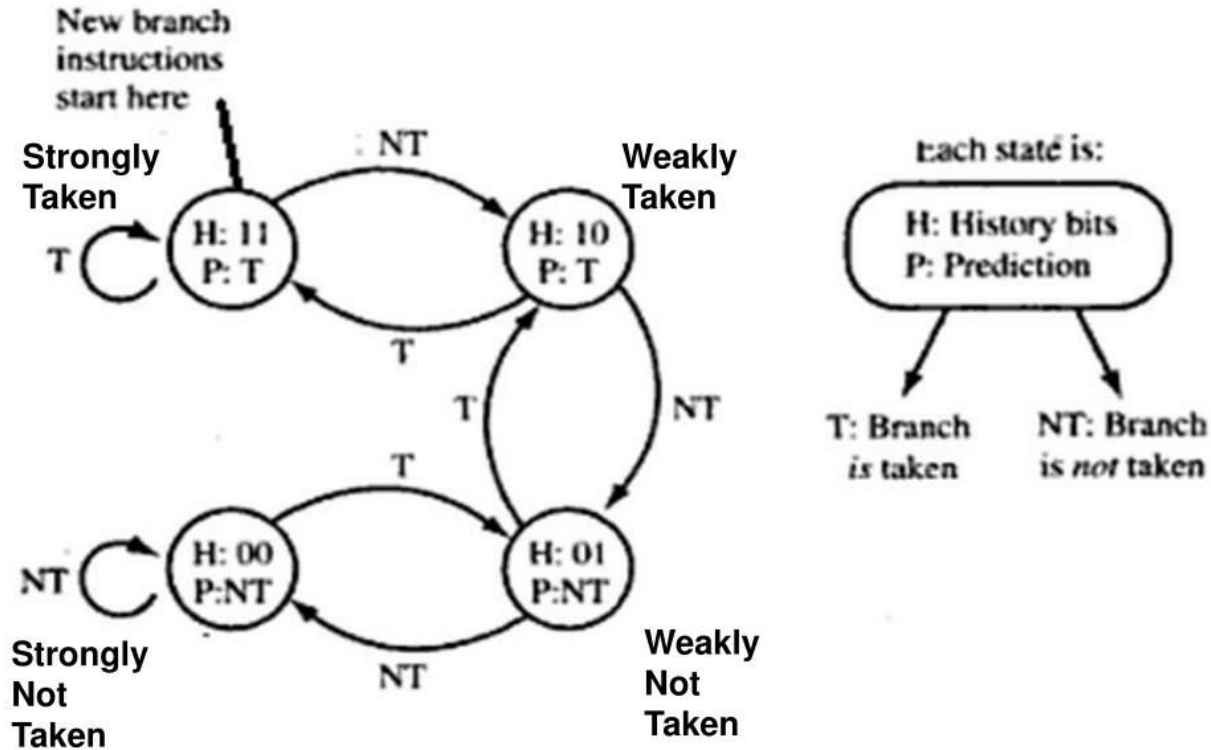
NT

T

STRONGLY
NOT
TAKEN

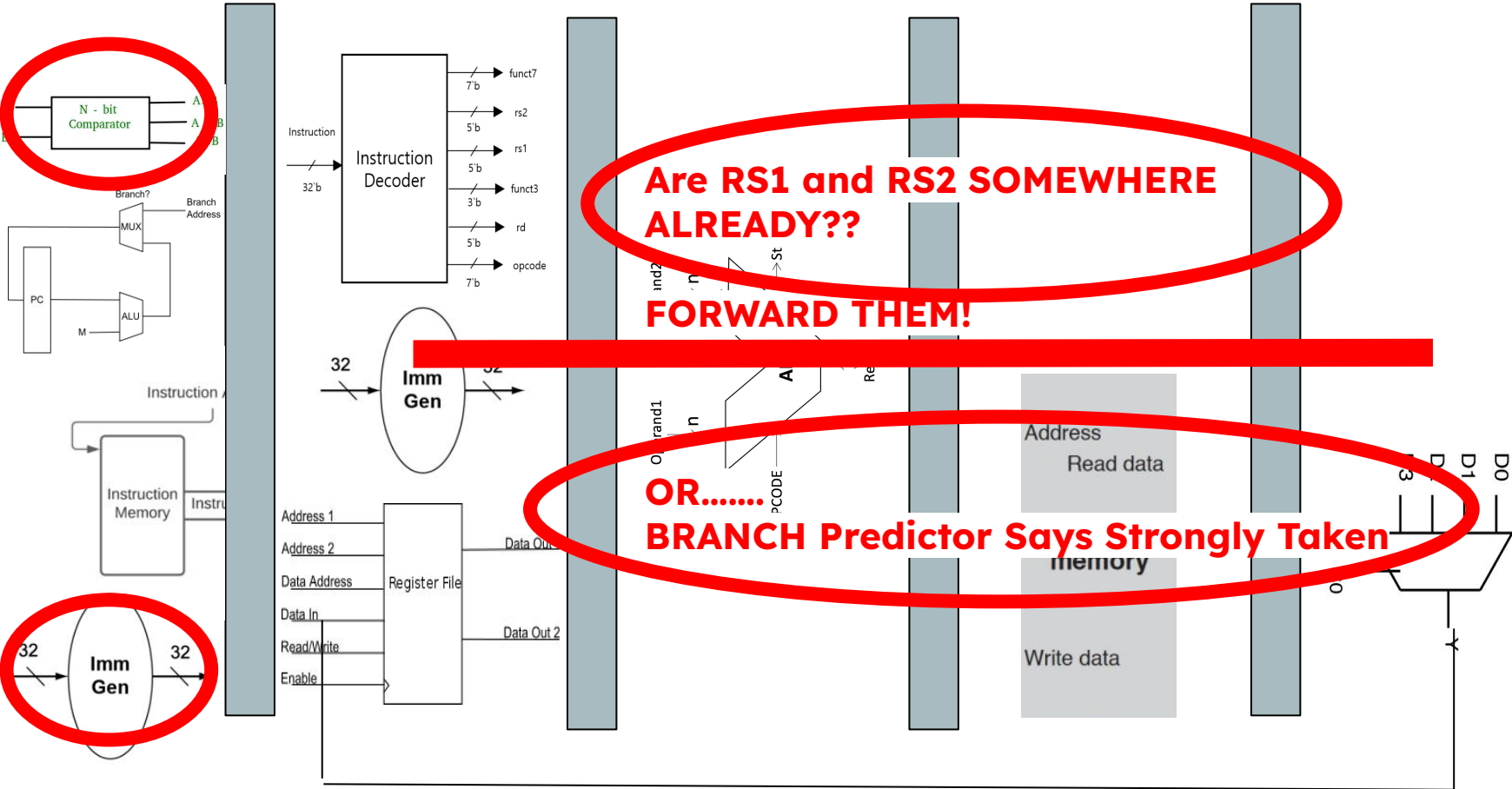
WEAKLY
NOT
TAKEN

Dynamic Branch Prediction Mechanism

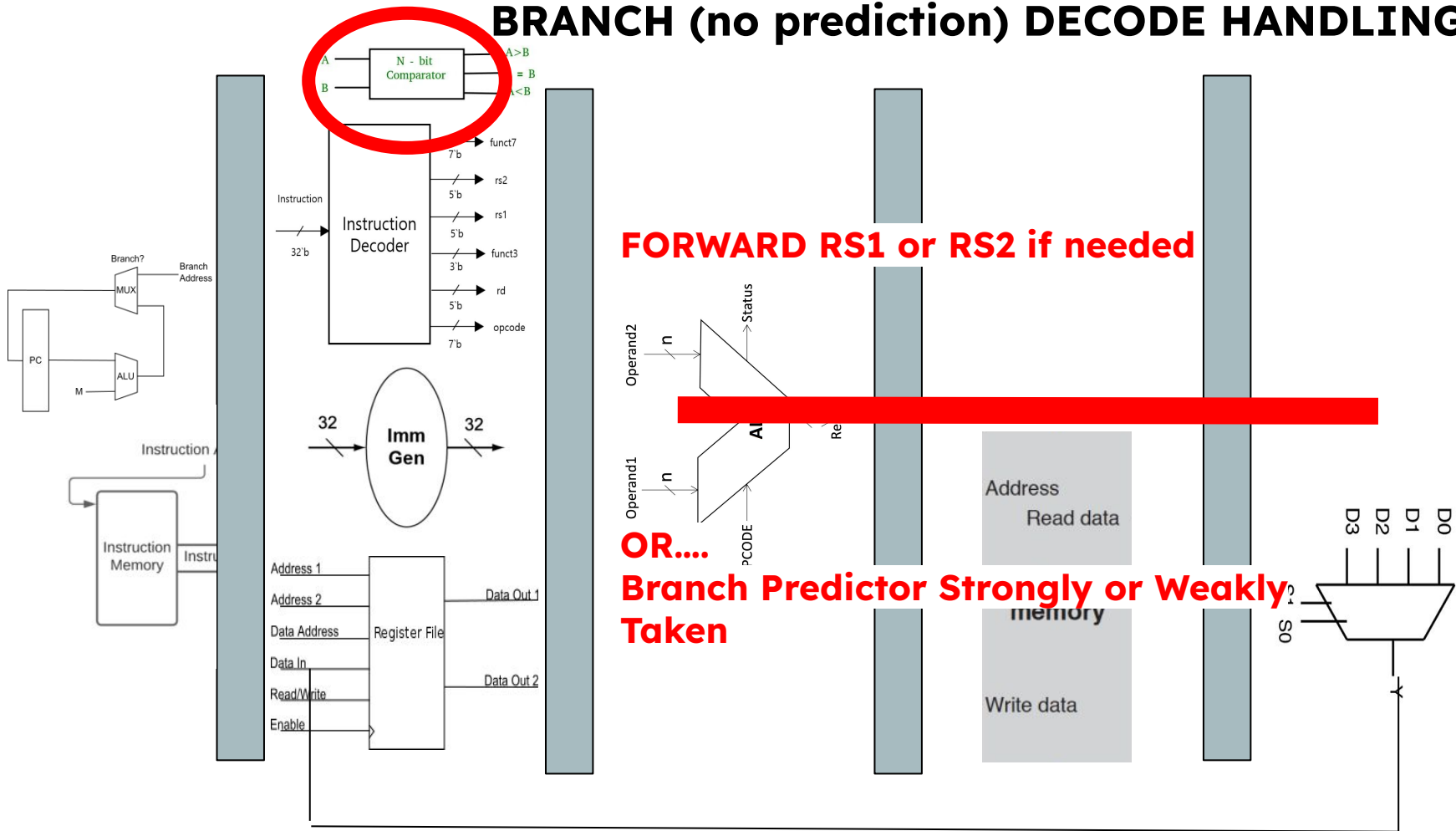


Revisiting Stage Handling for Branches (with Branch Prediction)

BRANCH Prediction FETCH HANDLING



BRANCH (no prediction) DECODE HANDLING



Lets do it again...

