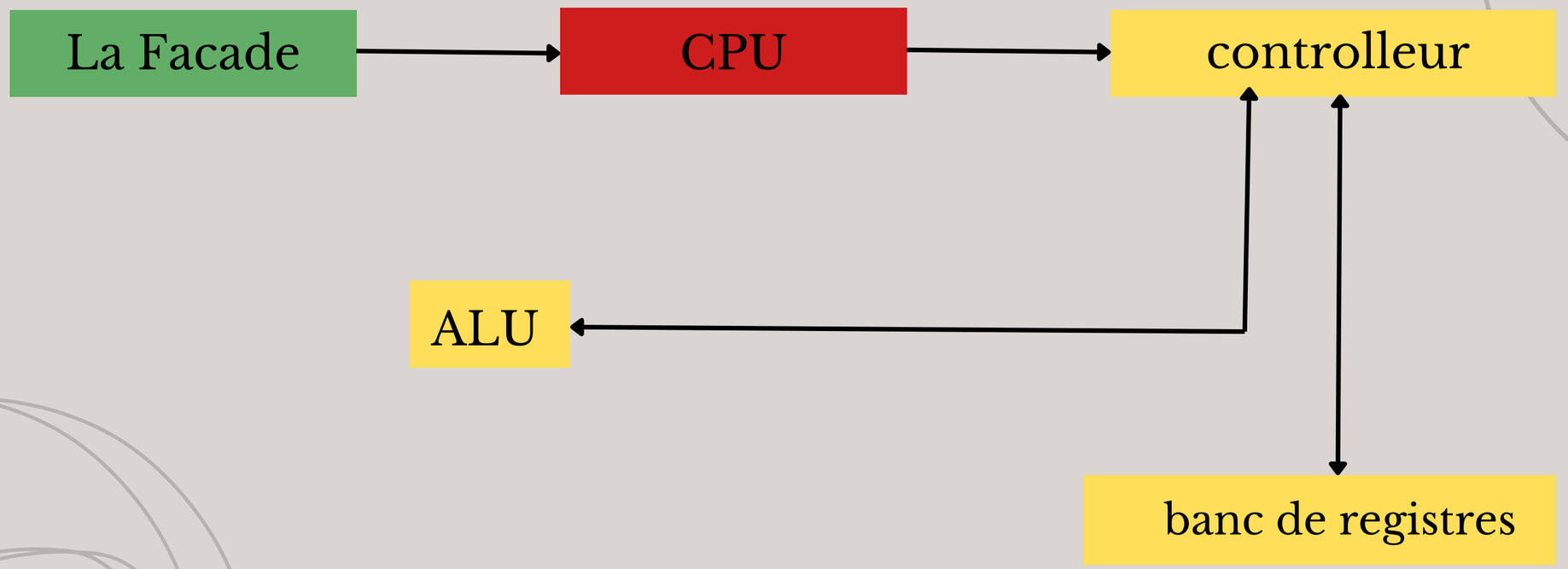
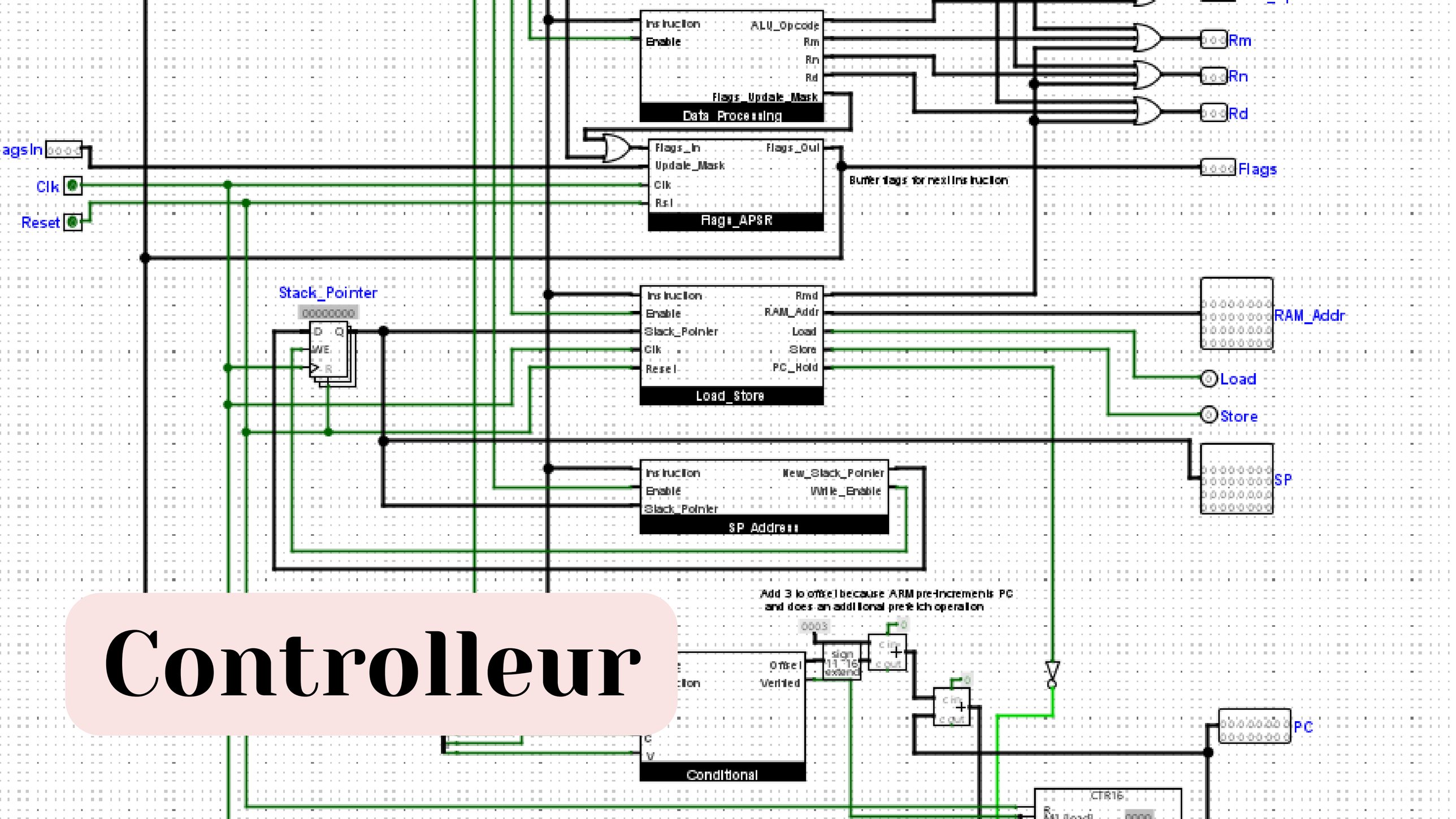
Polytech Nice Sophia

PARM

Ludovic Clolot
Dorian Renier
Mohamed Amine Youssef
Mohamed Rayene Romdhane

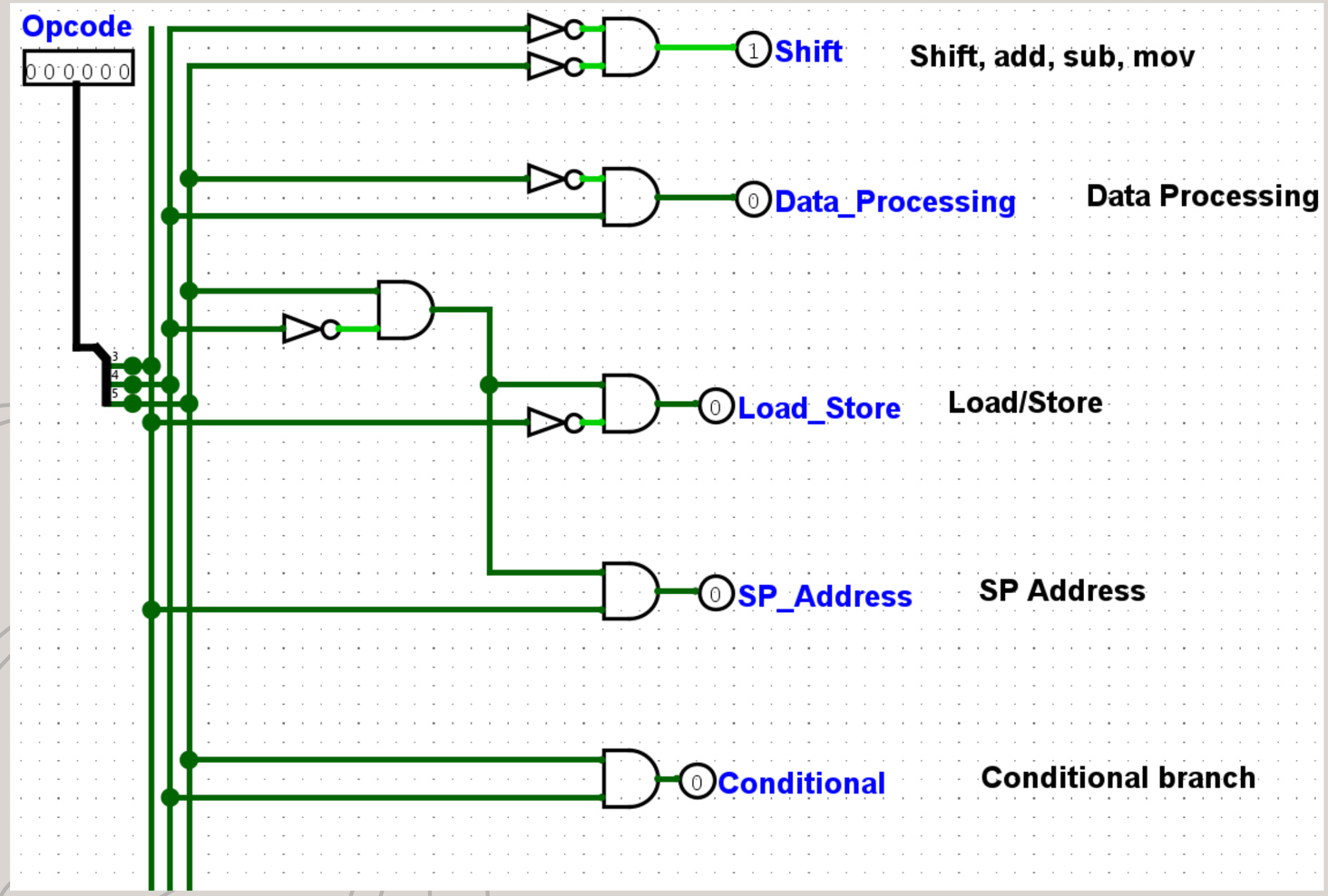
Architecture



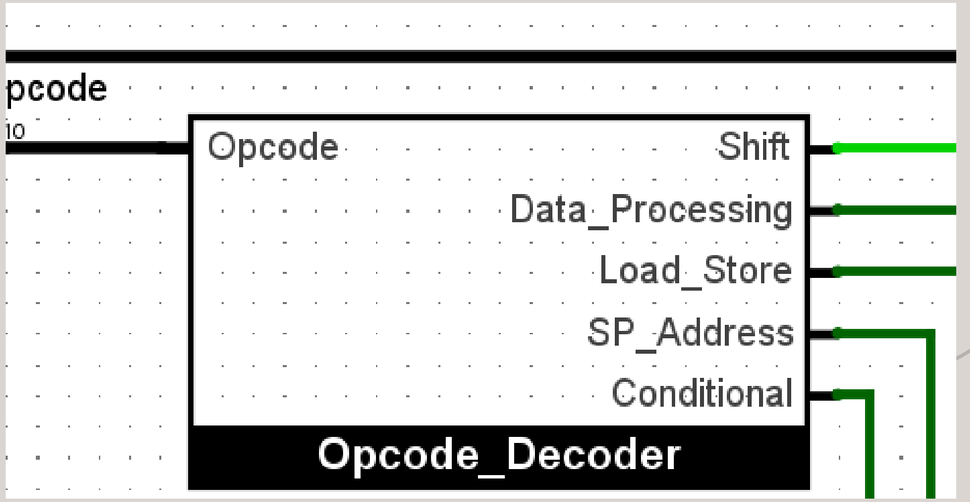


Controlleur

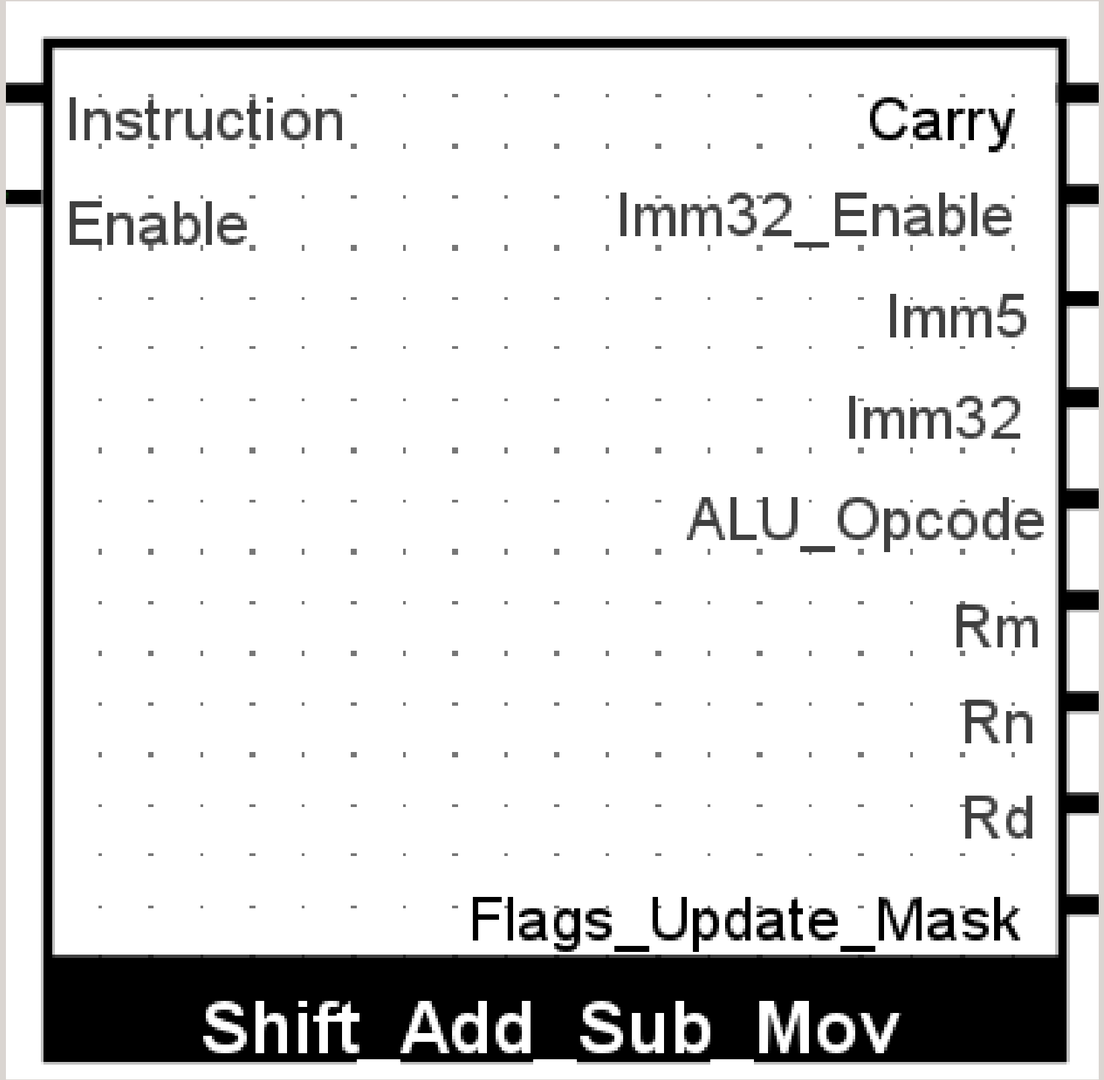
Opcode_Decoder



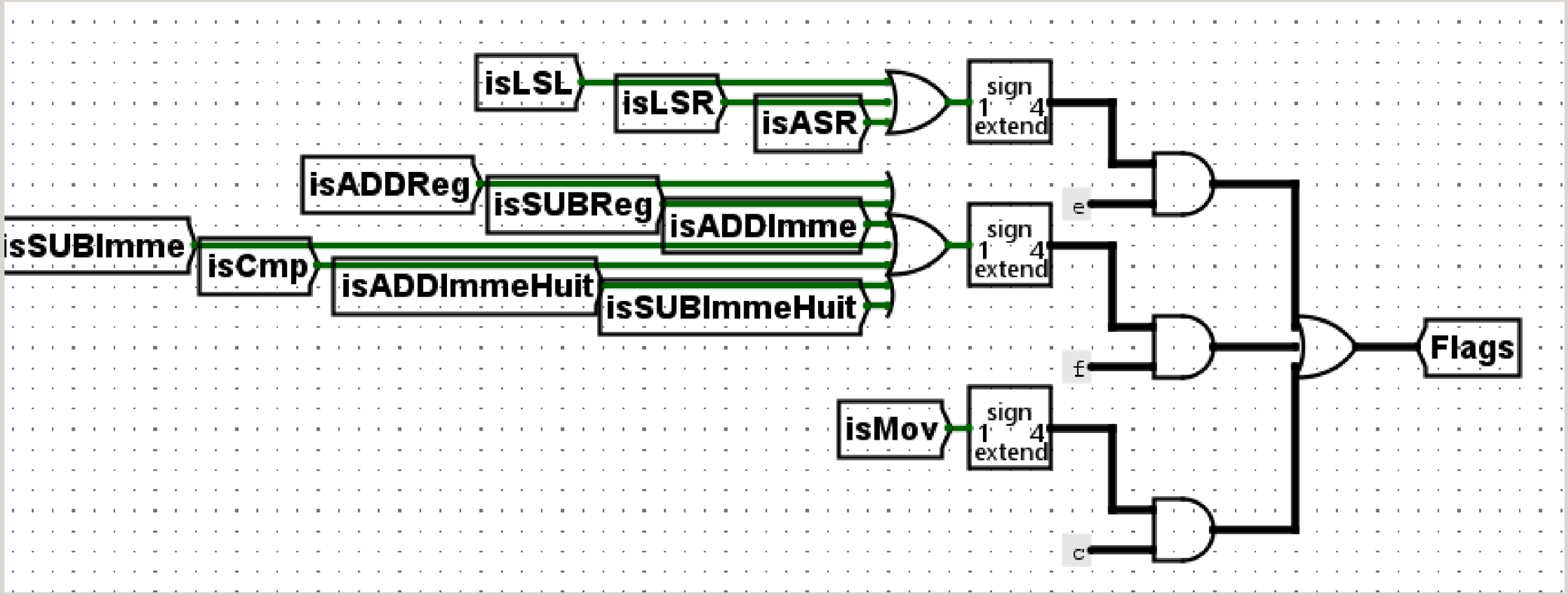
identification de l'instruction demandée



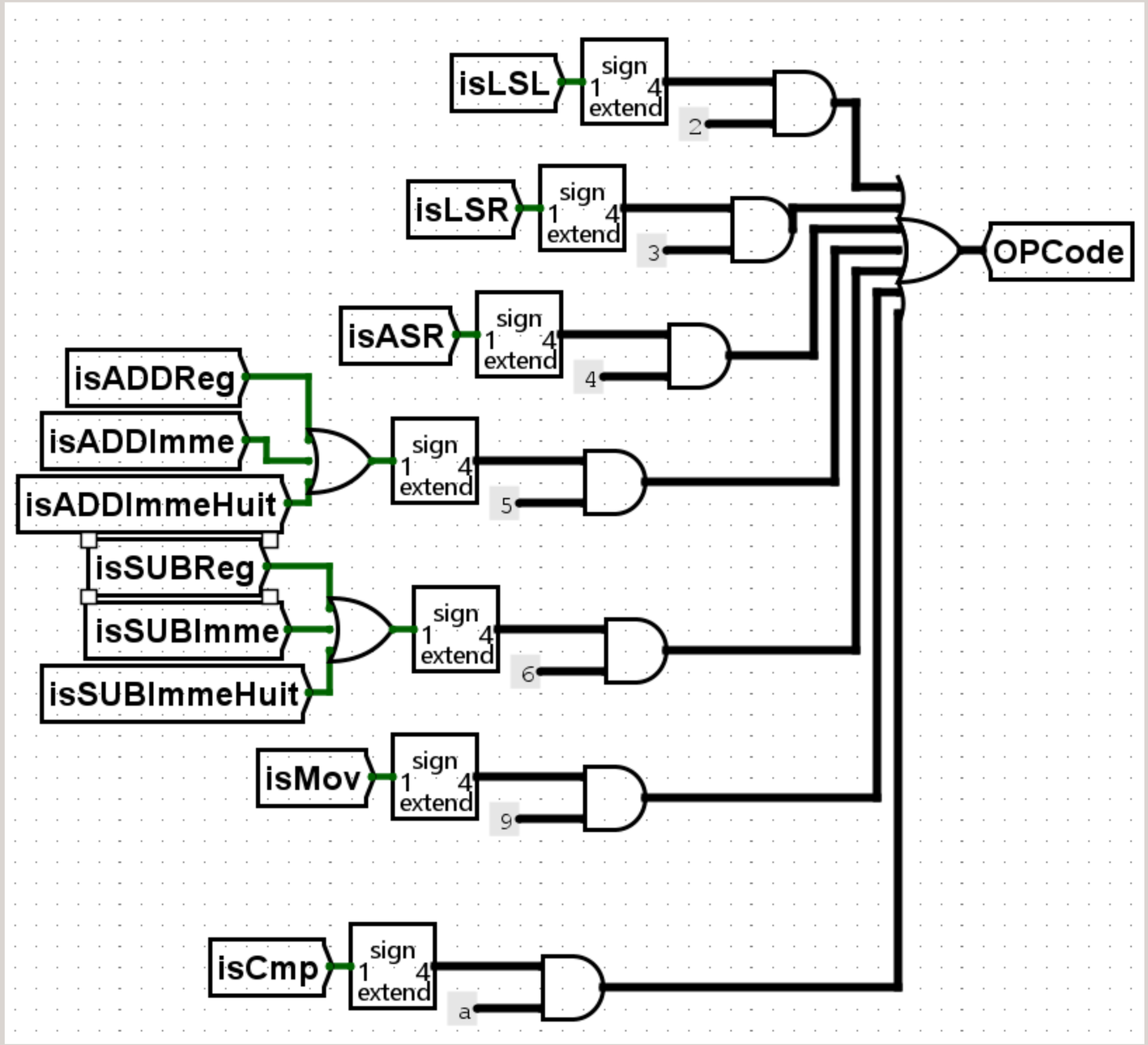
Shift, Add, Sub, Mov



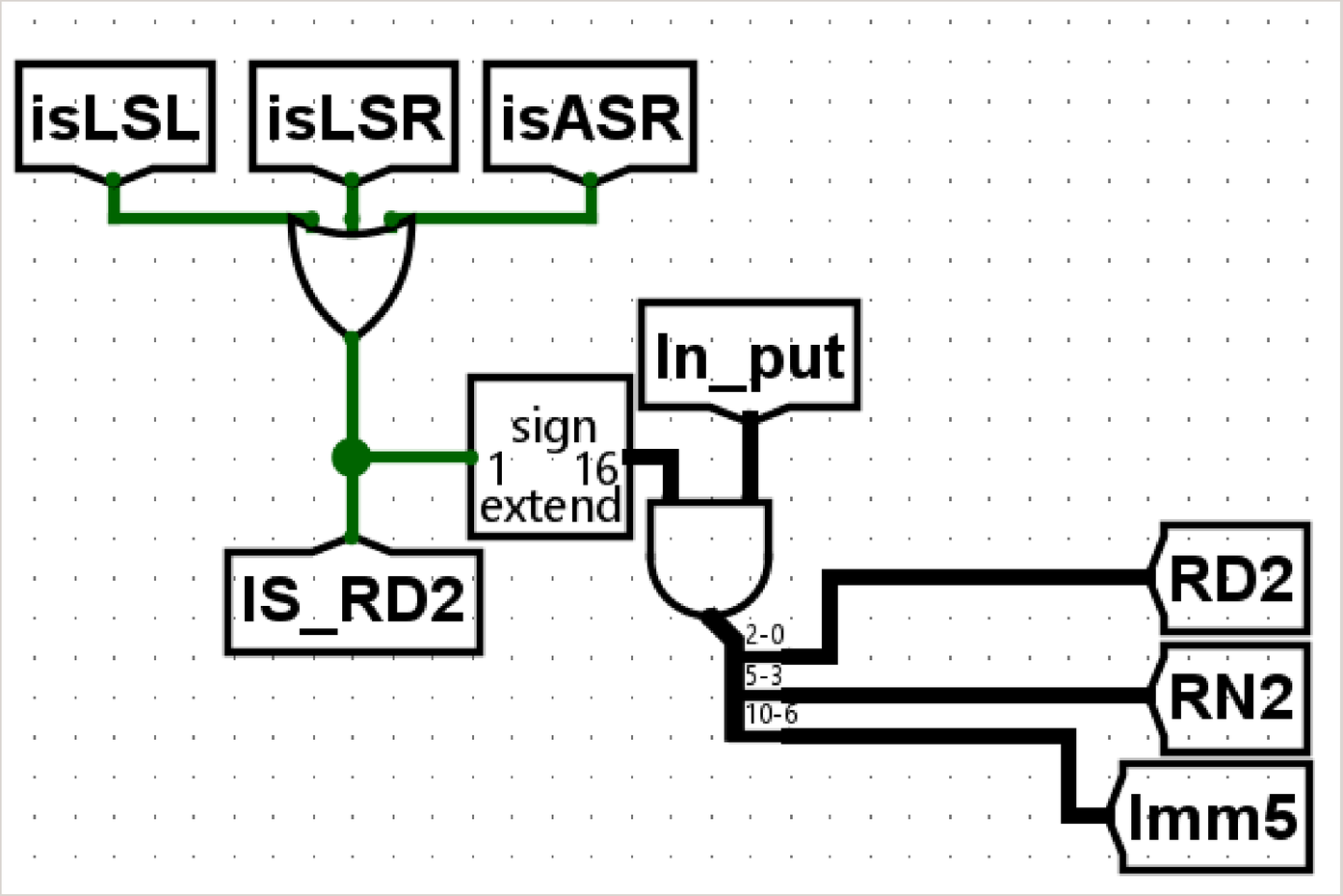
Calcul des flags



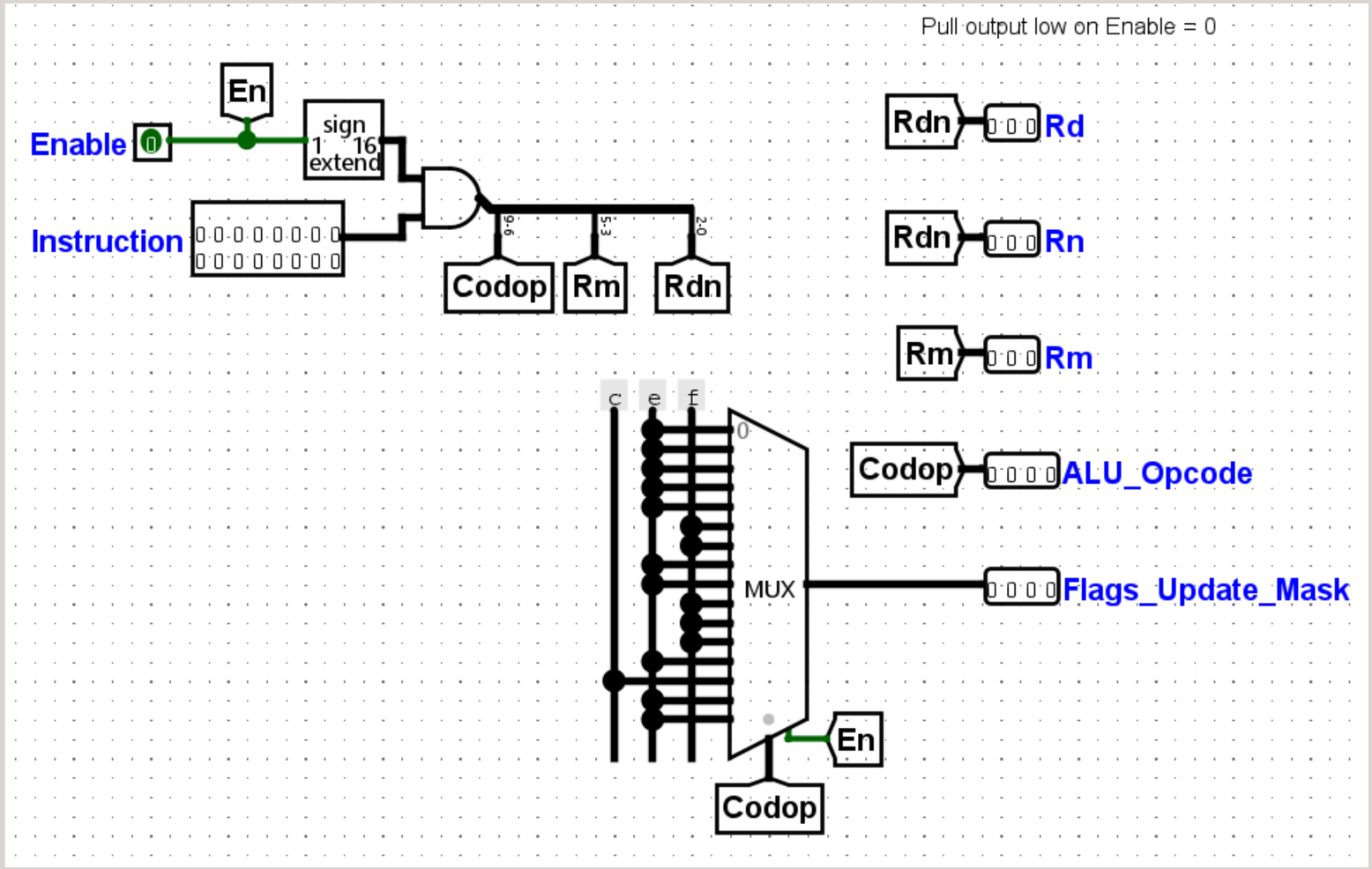
Détermination de l’ Opcode

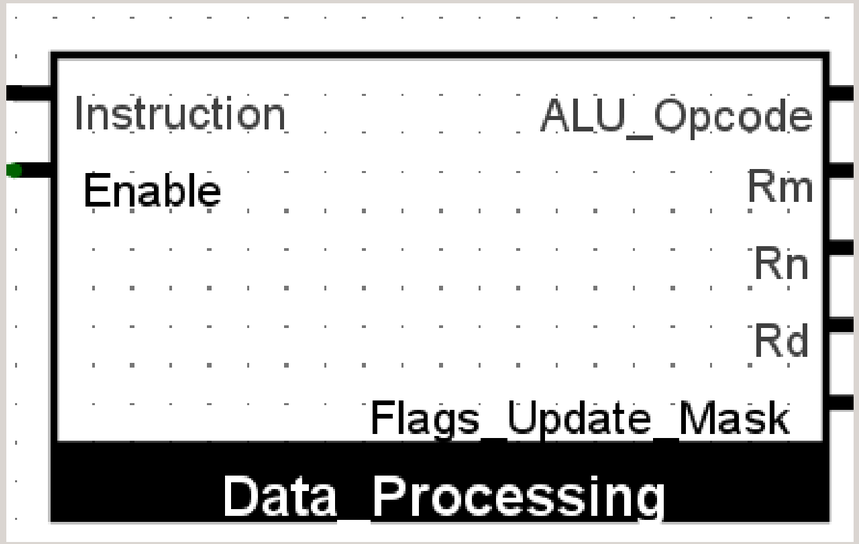


Traitement des addresses et immédiats

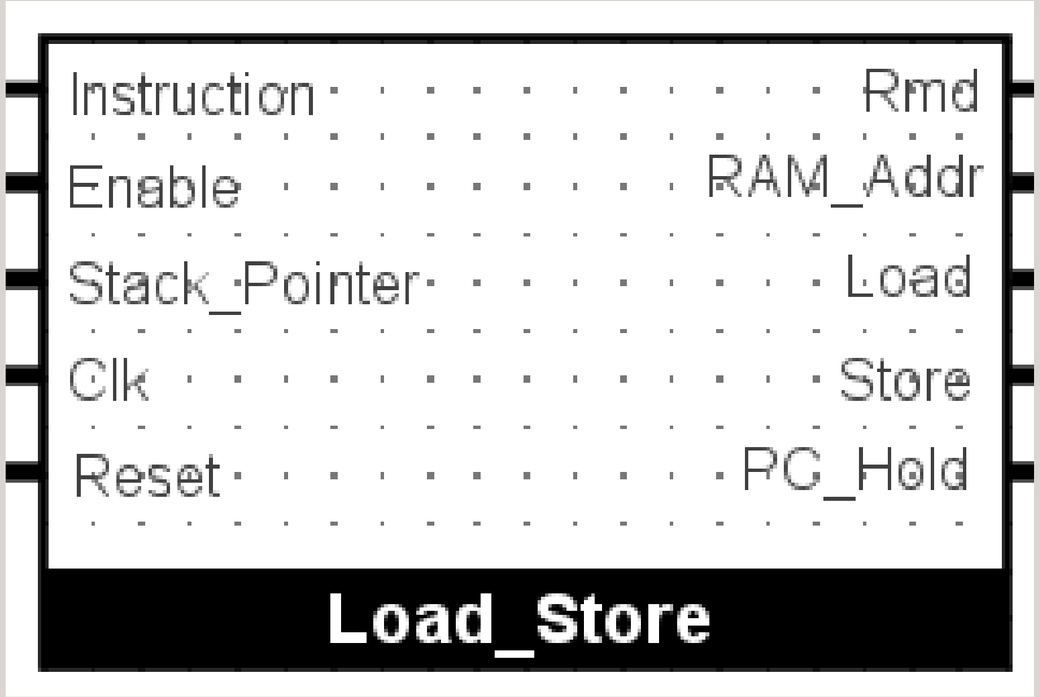


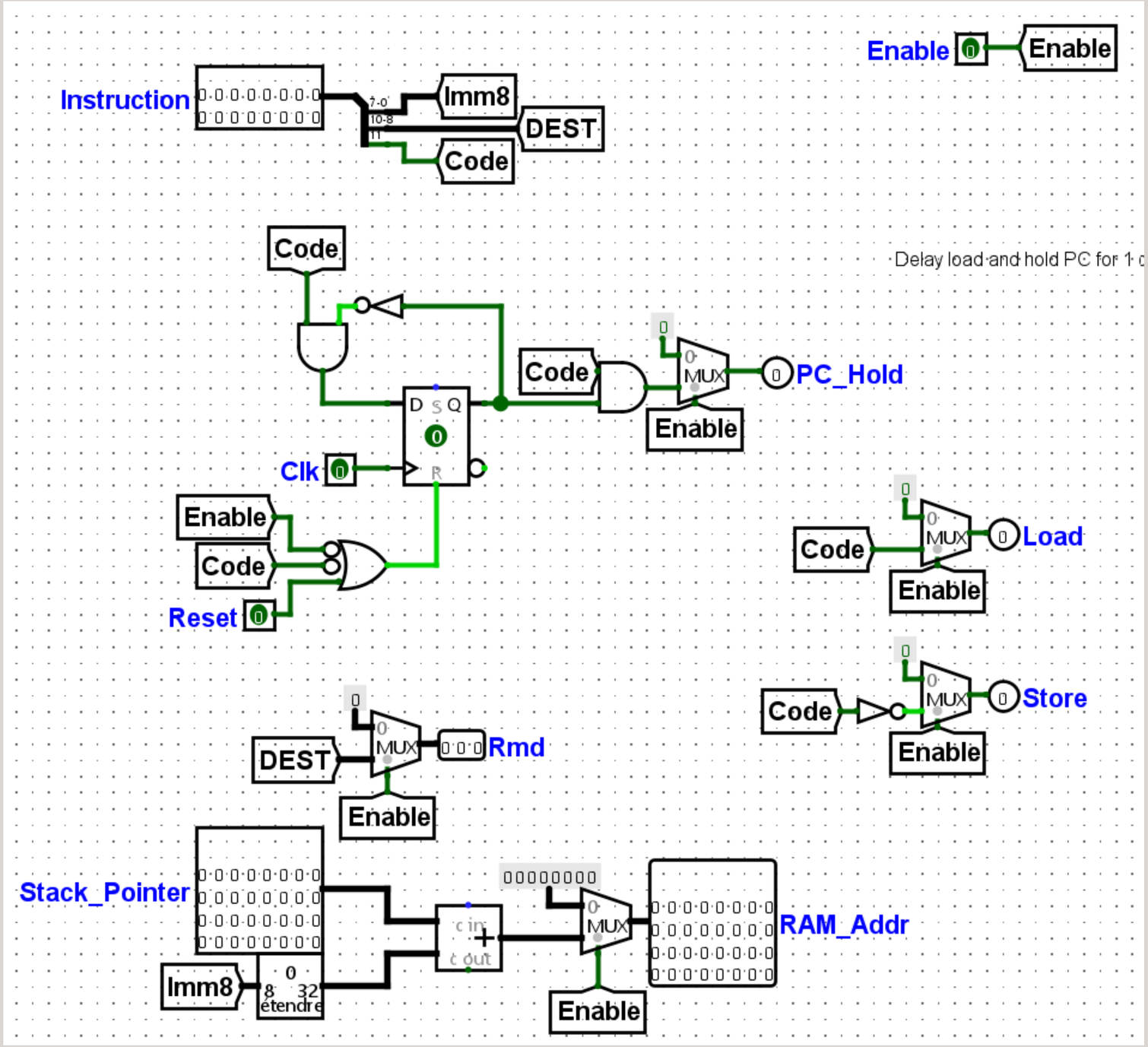
Data Processing



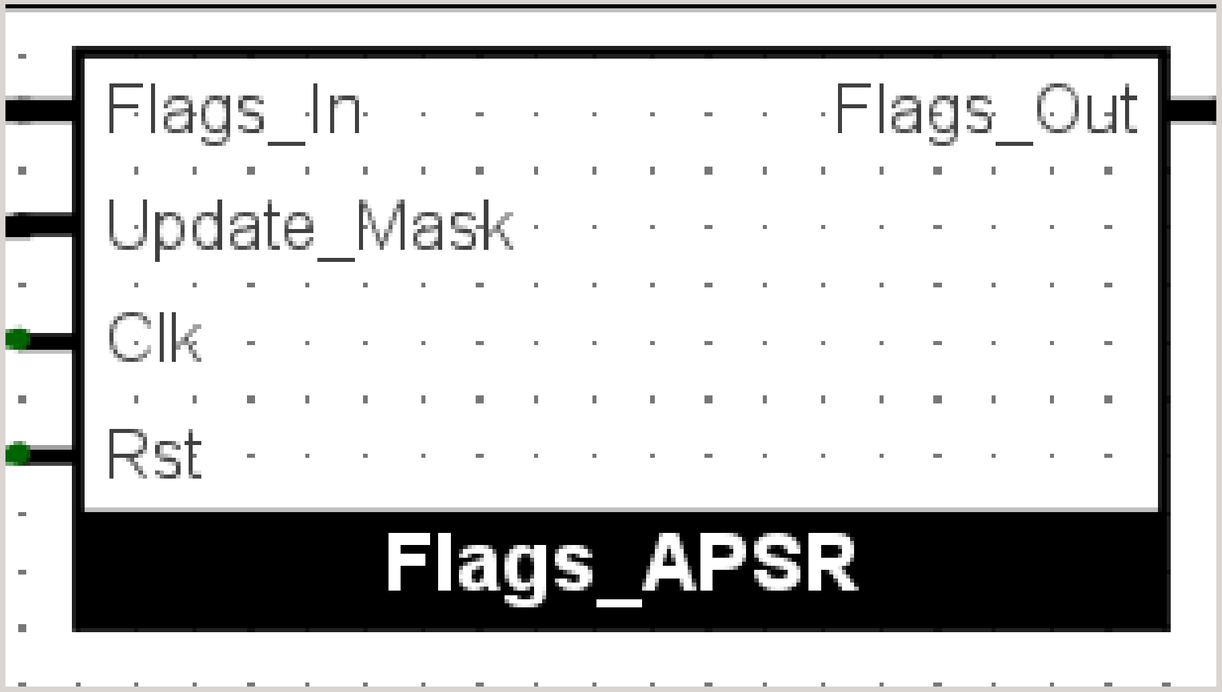


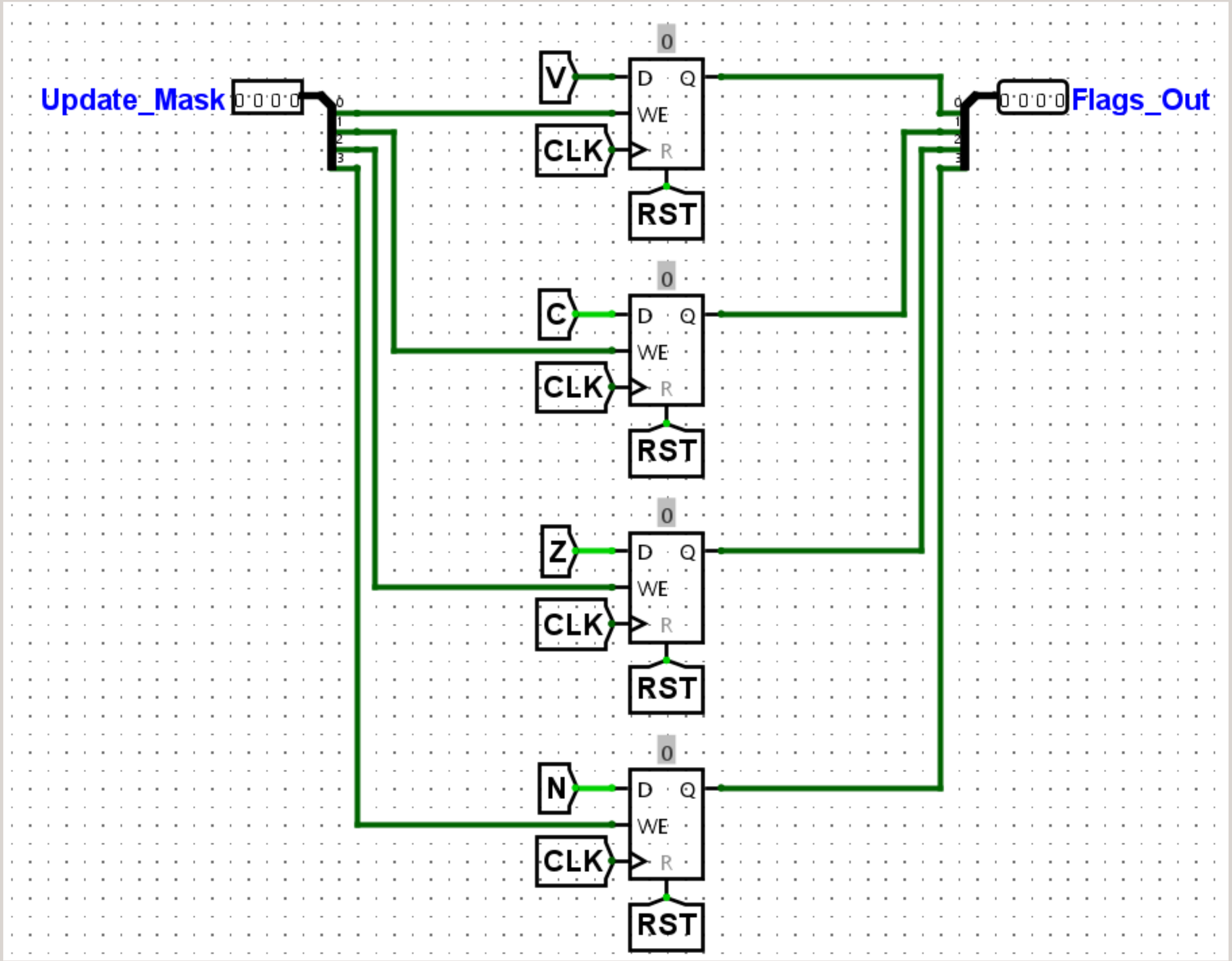
Load / Store


Instruction
Enable
Stack_Pointer
Clk
Reset
Rmd
RAM_Addr
Load
Store
PC_Hold
Load_Store

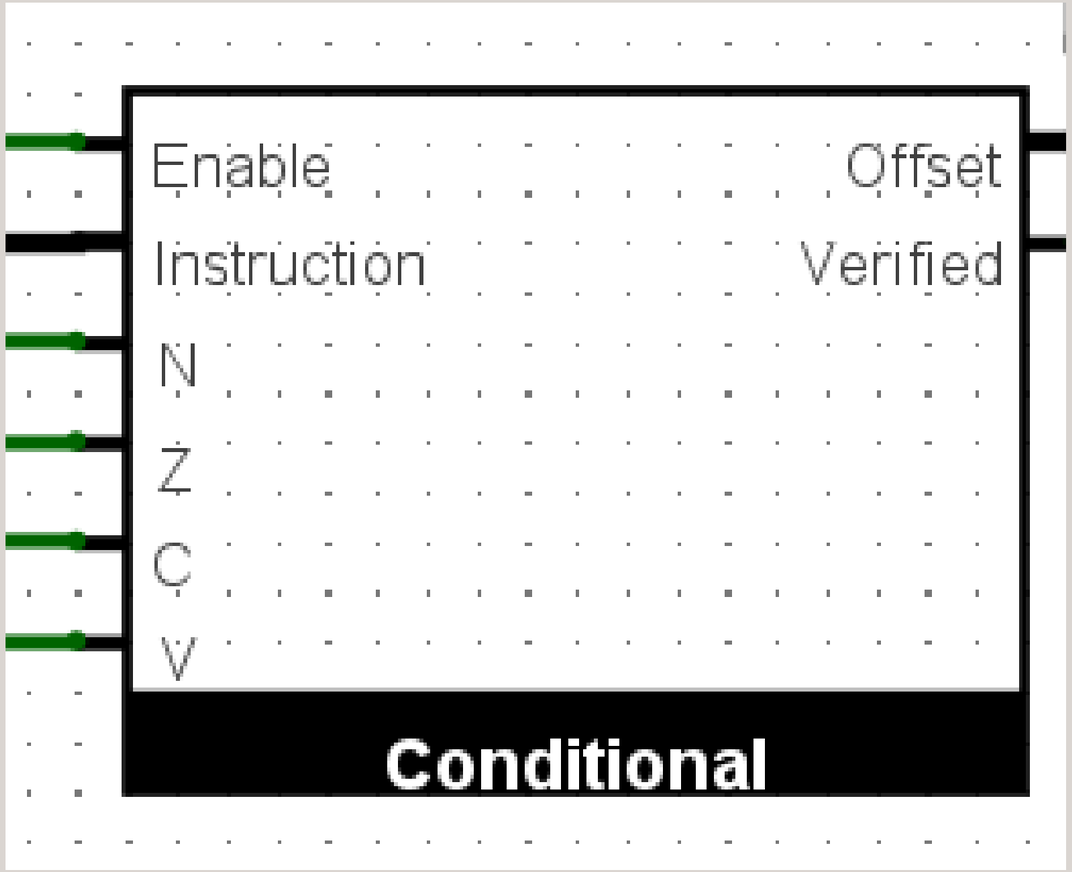

Enable
Enable
Instruction
7-0
10-8
11
Imm8
DEST
Code
Code
Delay load and hold PC for 1
Code
MUX
PC_Hold
Enable
D S Q
Clk
R
Enable
Code
Reset
MUX
Load
Code
Enable
MUX
Store
Code
Enable
DEST
MUX
Rmd
Enable
Stack_Pointer
c in
c out
MUX
RAM_Addr
Imm8
8
32
étendre
Enable

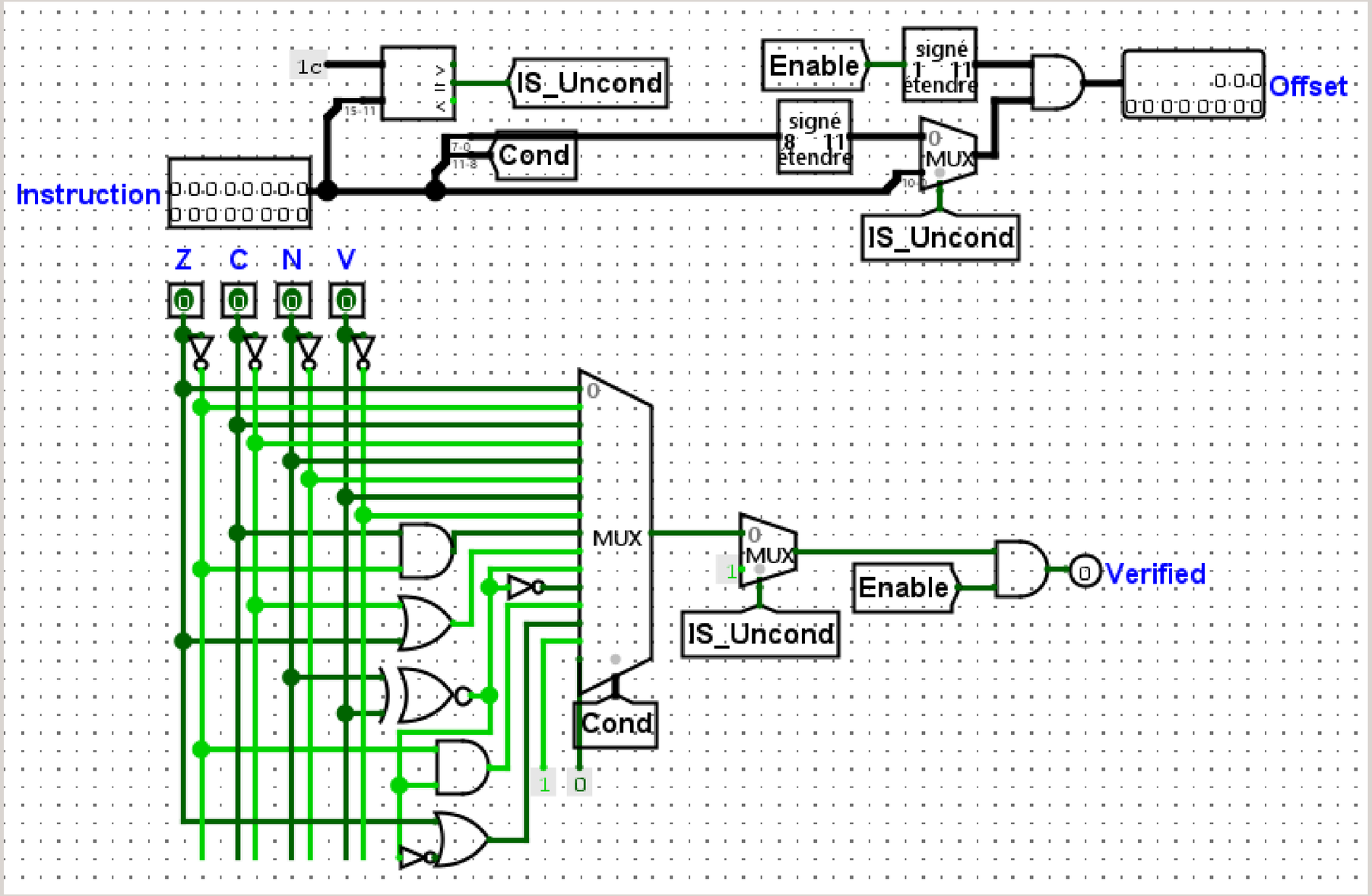
Flag ASRP




Update_Mask
0 0 0 0
0
1
2
3
V
C
Z
N
CLK
RST
D
Q
WE
R
0
Flags_Out

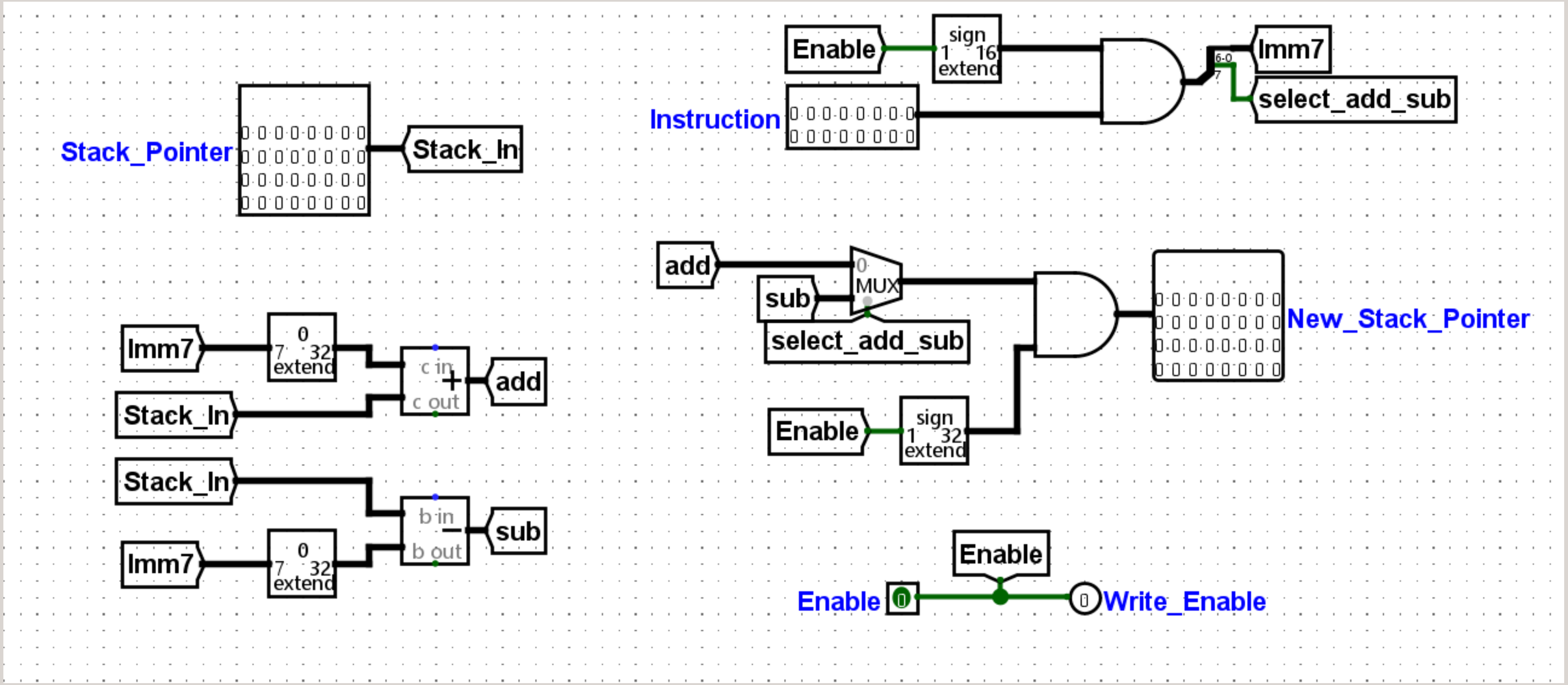
Conditional



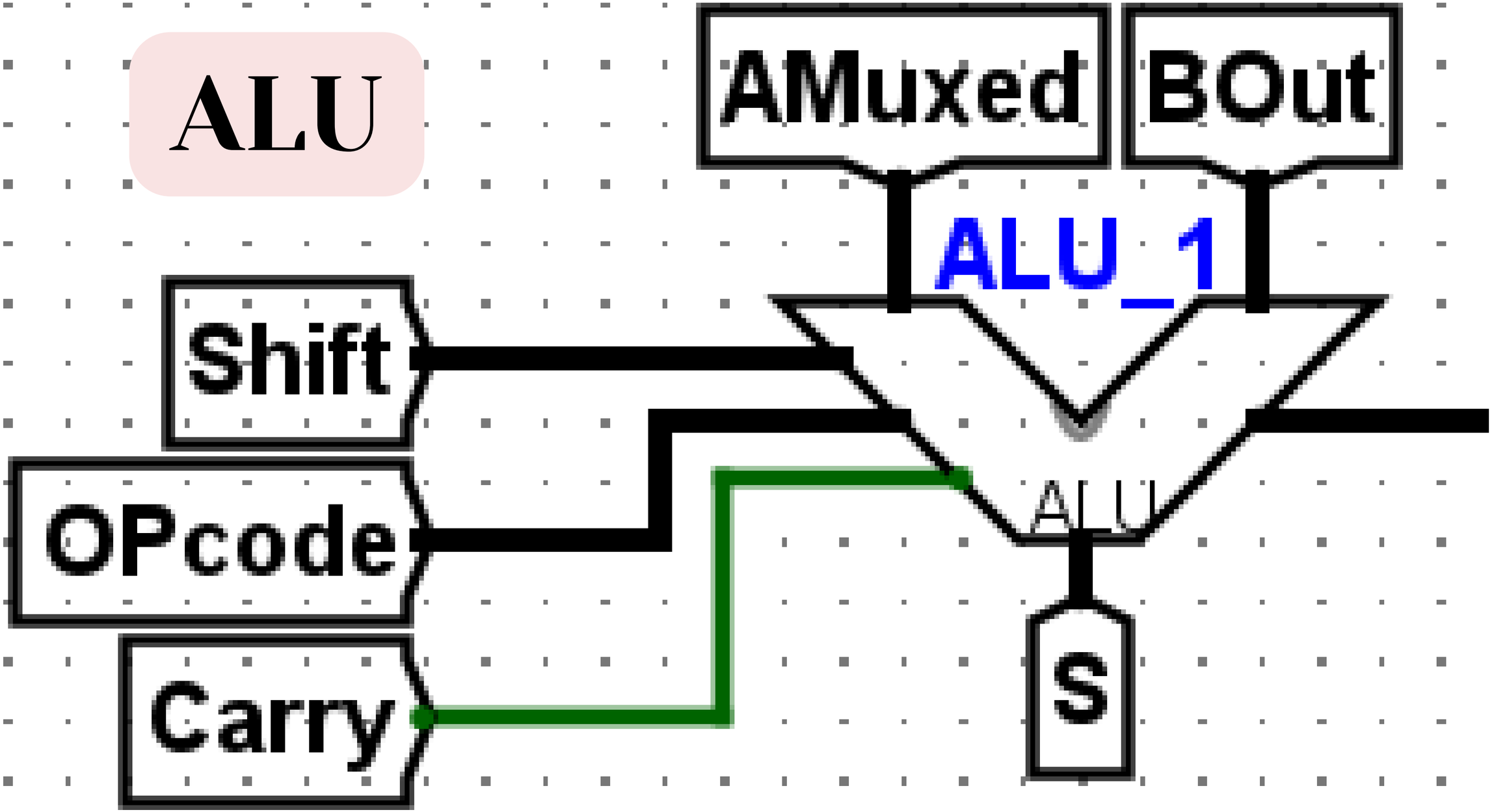

1c
15-11
IS_Uncond
Enable
signé
étendre
Offset
Instruction
7-0
11-8
Cond
signé
étendre
MUX
IS_Uncond
Z
C
N
V
MUX
MUX
IS_Uncond
Enable
Verified
Cond

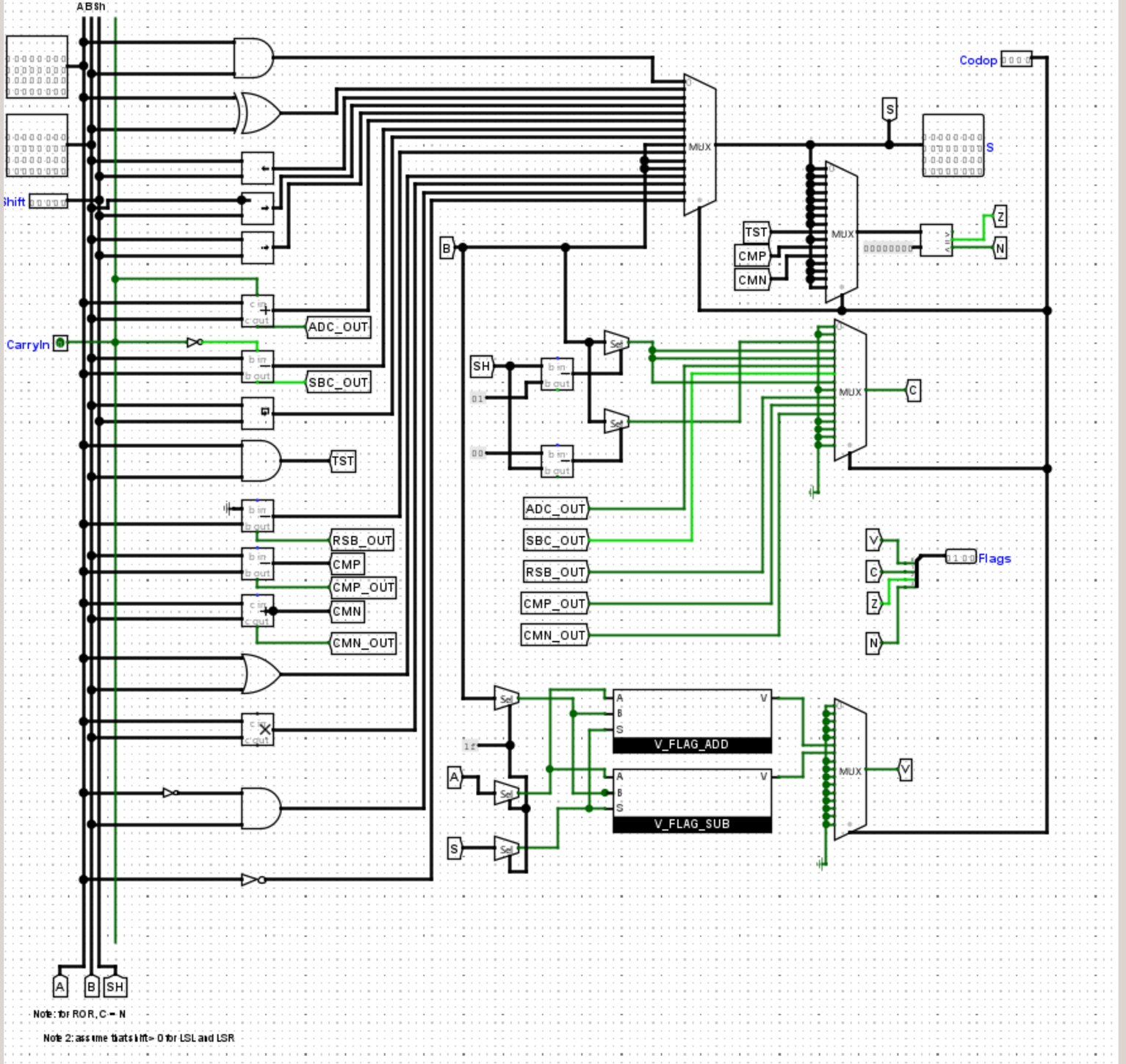
SP_Address

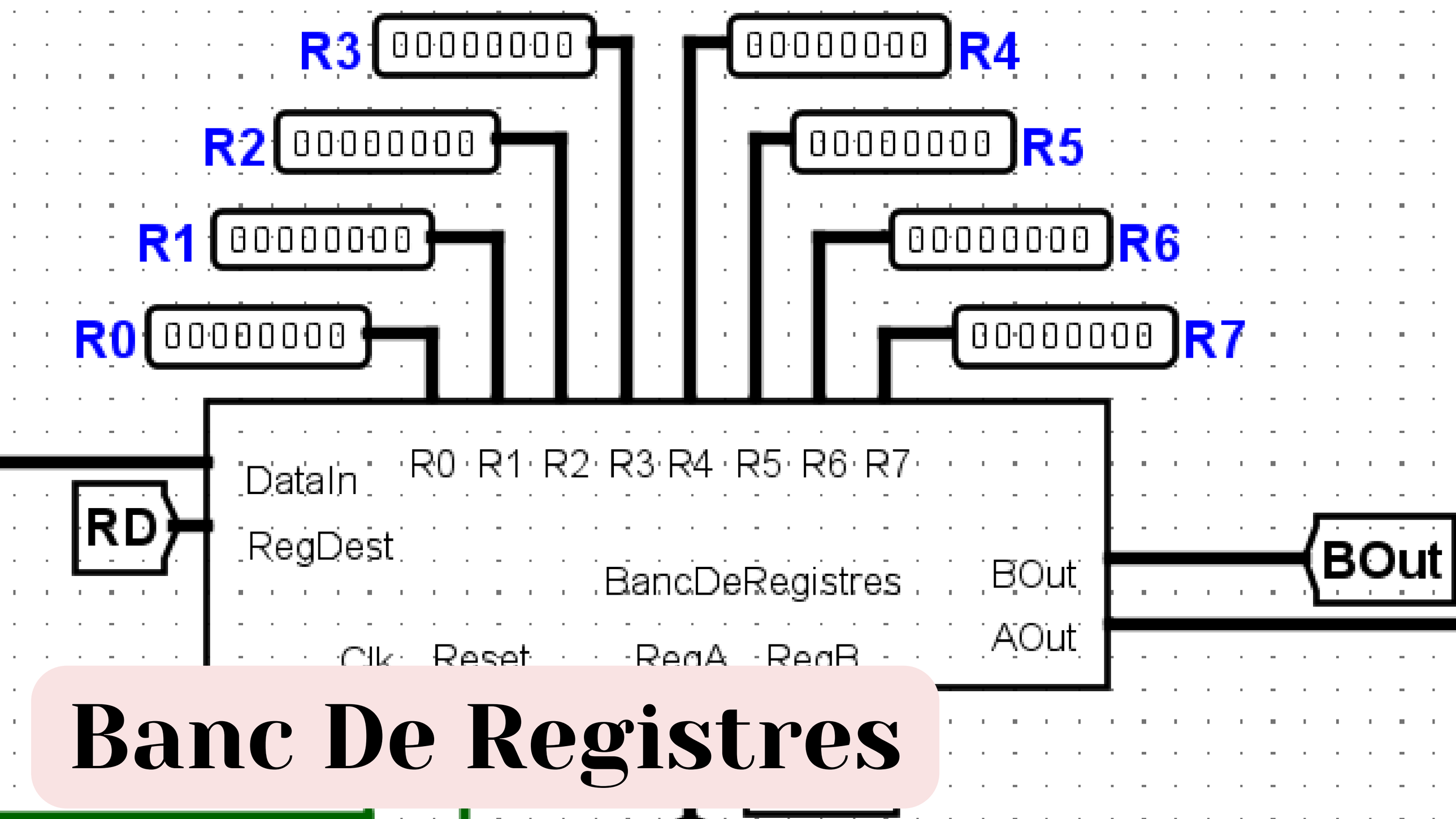

Instruction
Enable
Stack_Pointer
New_Stack_Pointer
Write_Enable
SP_Address


Stack_Pointer
Stack_In
Enable
sign
1
16
extend
Instruction
6-0
7
Imm7
select_add_sub
add
0
MUX
sub
select_add_sub
New_Stack_Pointer
Enable
sign
1
32
extend
Imm7
0
7
32
extend
c in
+
c out
add
Stack_In
Stack_In
b in
−
b out
sub
Imm7
0
7
32
extend
Enable
Enable
0
0
Write_Enable

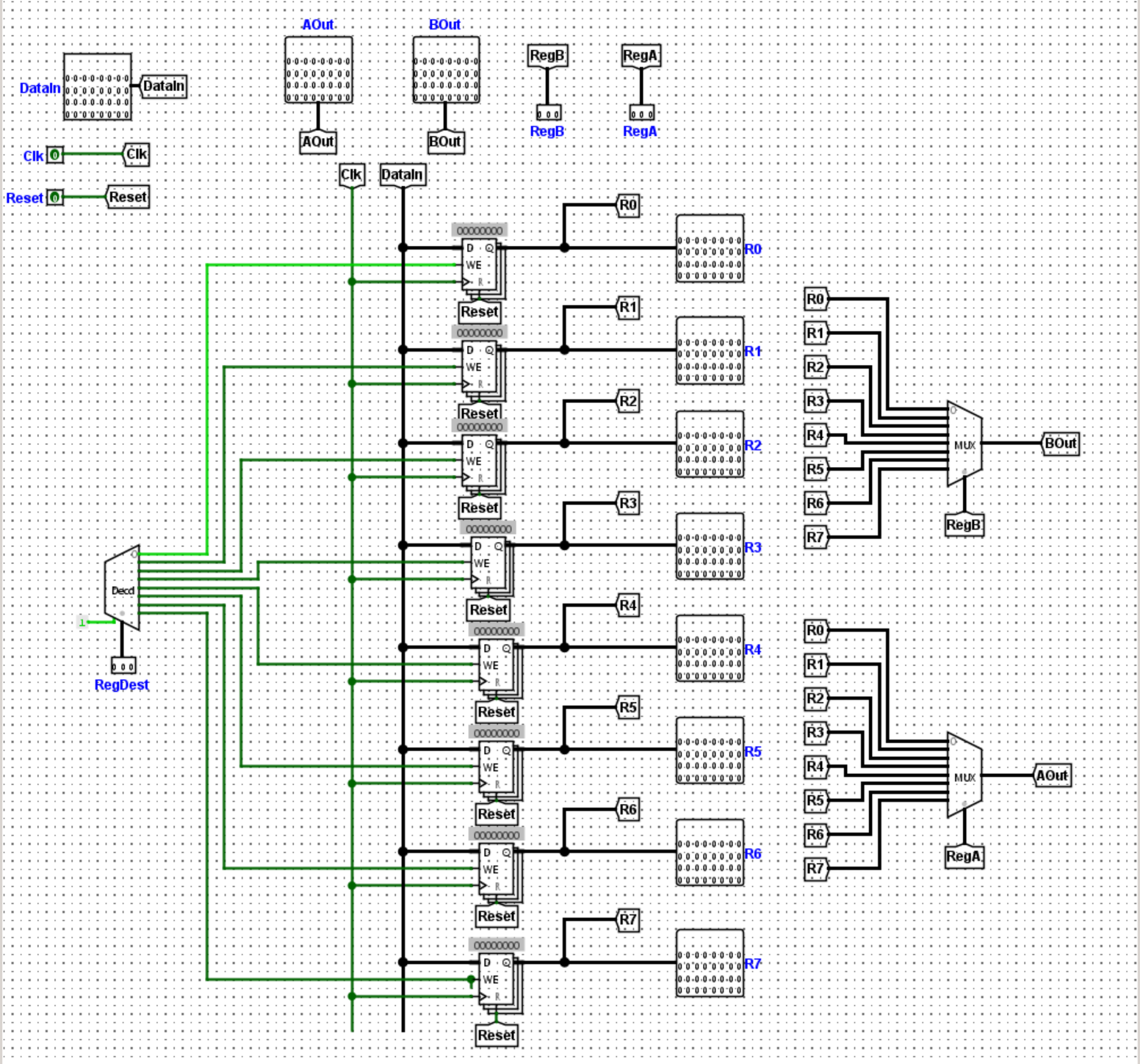
ALU

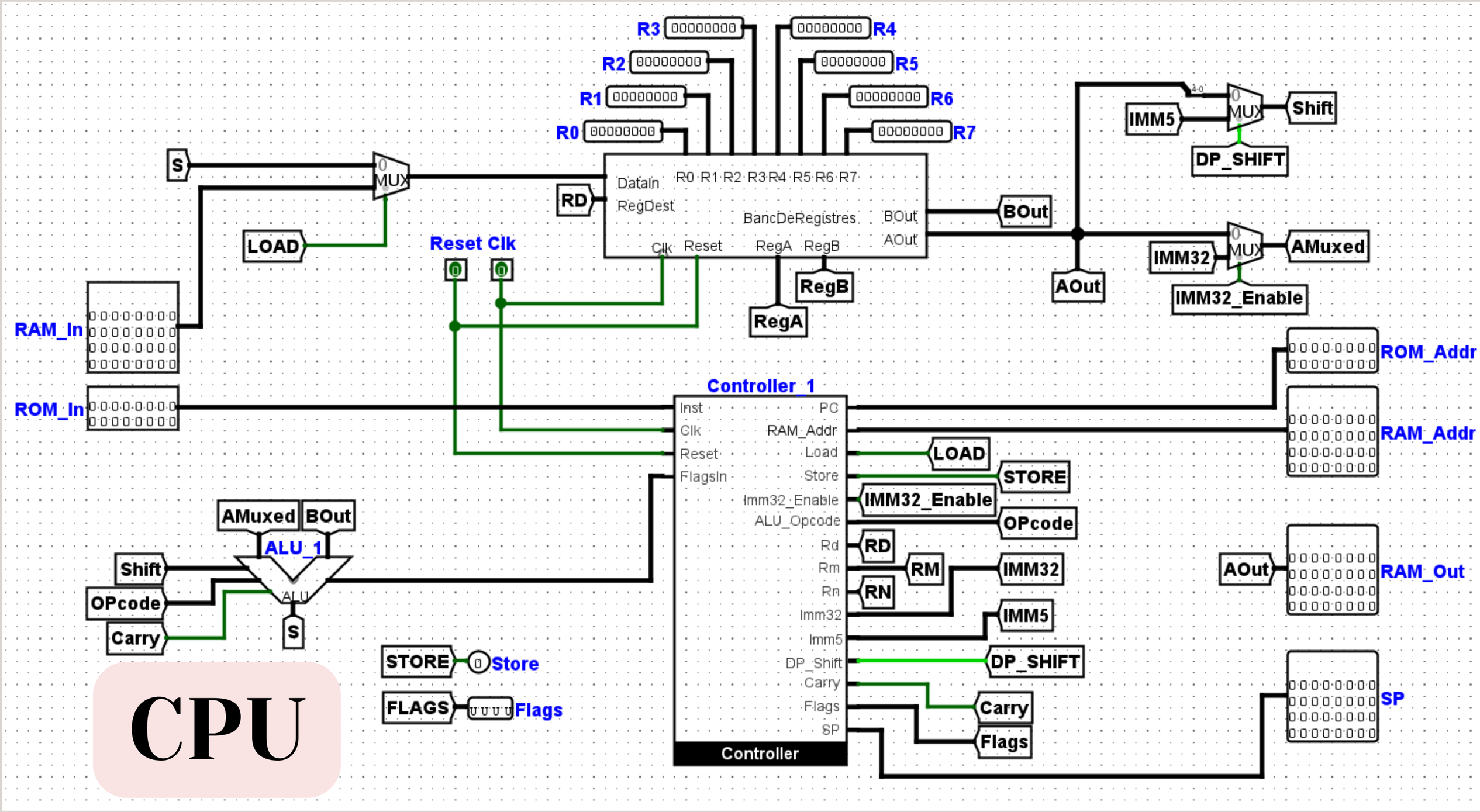



ABSh
Codop
Shift
CarryIn
ADC_OUT
SBC_OUT
TST
RSB_OUT
CMP
CMP_OUT
CMN
CMN_OUT
MUX
S
TST
CMP
CMN
Z
N
B
SH
C
V
Flags
A
V_FLAG_ADD
V_FLAG_SUB
Note: for ROR, C = N
Note 2: assume that shift= 0 for LSL and LSR

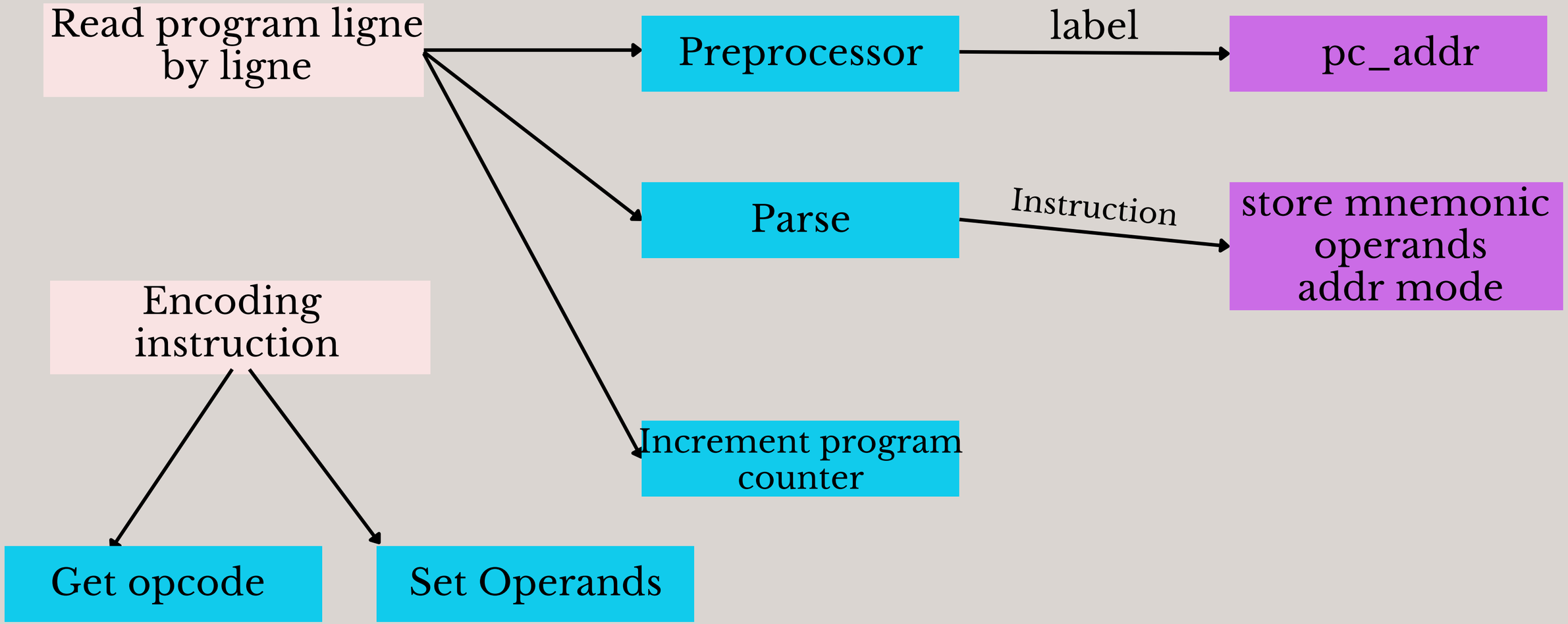


Banc De Registres


DataIn
DataIn
Clk
Clk
Reset
Reset
AOut
AOut
BOut
BOut
RegB
RegB
RegA
RegA
Clk
DataIn
Decd
1
RegDest
D
Q
WE
R
Reset
R0
R1
R2
R3
R4
R5
R6
R7
MUX
BOut
RegB
AOut
RegA


CPU
R0
R1
R2
R3
R4
R5
R6
R7
S
MUX
LOAD
RD
DataIn
RegDest
BancDeRegistres
BOut
AOut
Clk
Reset
RegA
RegB
Reset Clk
RAM_In
ROM_In
IMM5
Shift
DP_SHIFT
IMM32
AMuxed
IMM32_Enable
BOut
AOut
Controller_1
Inst
Clk
Reset
FlagsIn
PC
RAM_Addr
Load
Store
Imm32_Enable
ALU_Opcode
Rd
Rm
Rn
Imm32
Imm5
DP_Shift
Carry
Flags
SP
Controller
ROM_Addr
RAM_Addr
STORE
IMM32_Enable
OPcode
RM
RN
IMM32
IMM5
DP_SHIFT
Carry
Flags
AOut
RAM_Out
SP
AMuxed
BOut
ALU_1
ALU
Shift
OPcode
Carry
S
STORE
Store
FLAGS
Flags

Assembleur


Read program ligne by ligne
Preprocessor
label
pc_addr
Parse
Instruction
store mnemonic operands addr mode
Encoding instruction
Increment program counter
Get opcode
Set Operands

DEMO