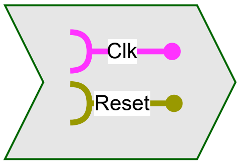
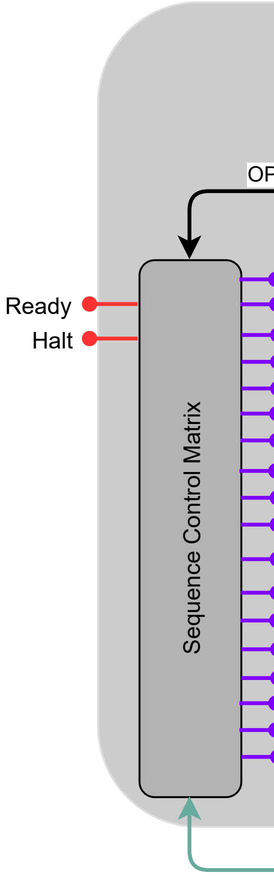
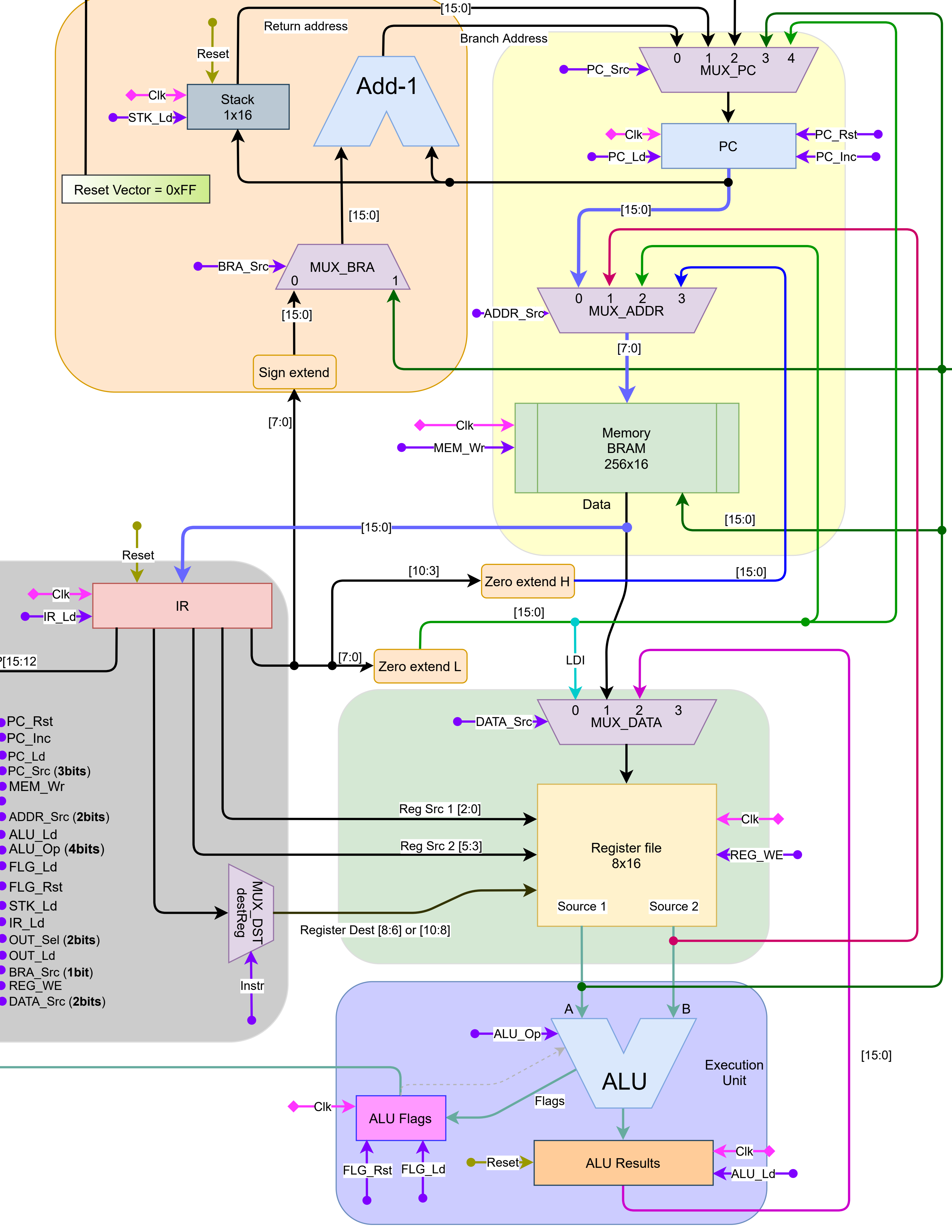
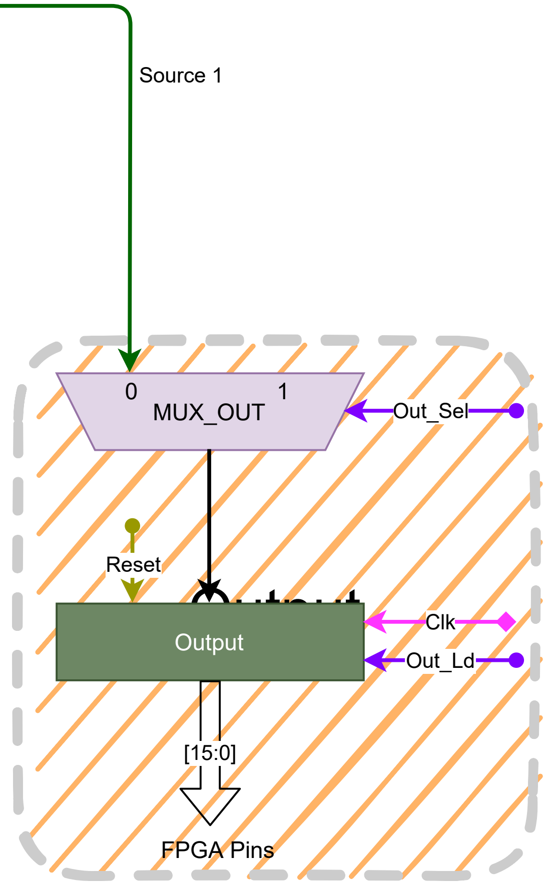
000
001
010
011
100

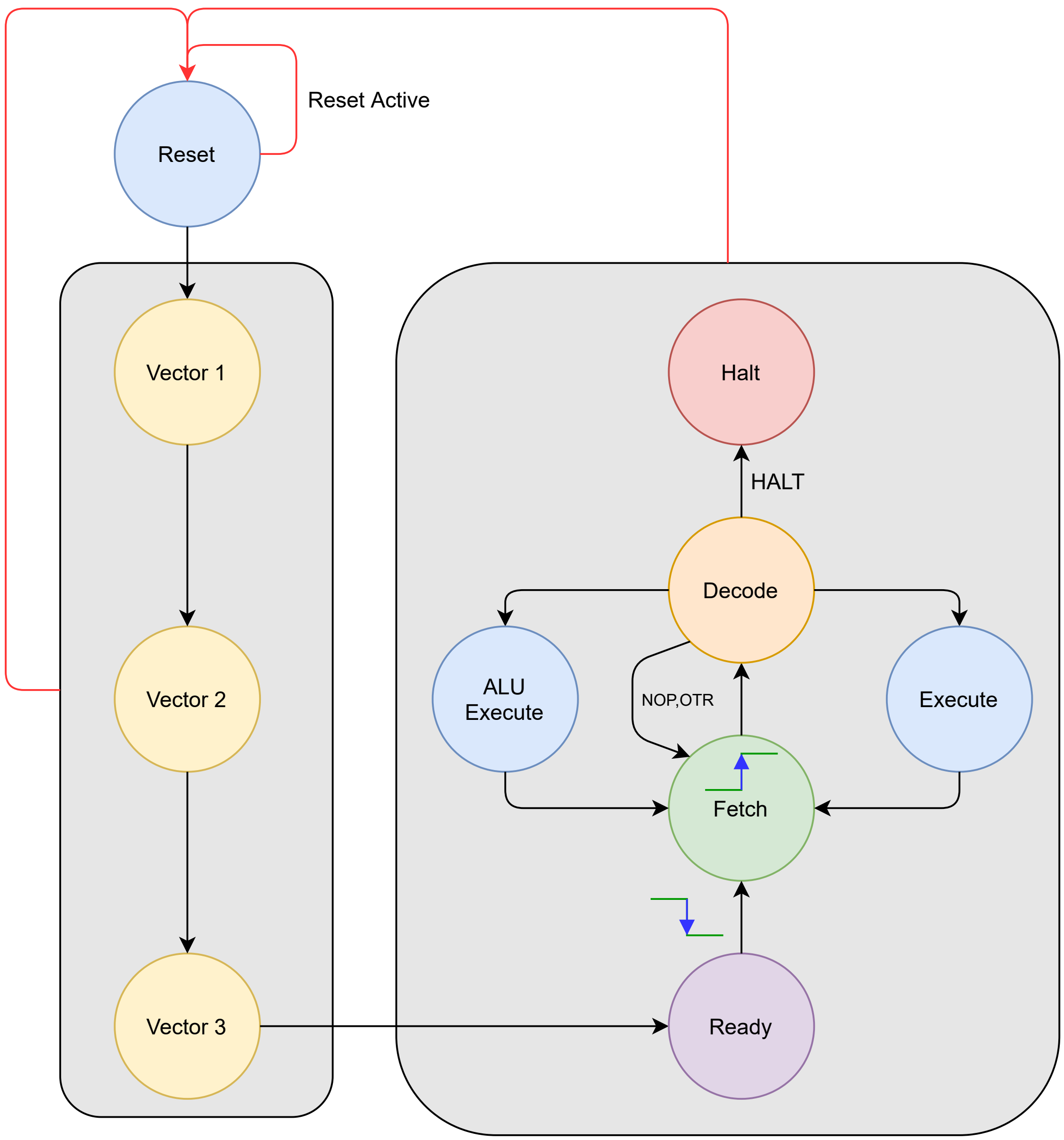
--- Version 3 ---

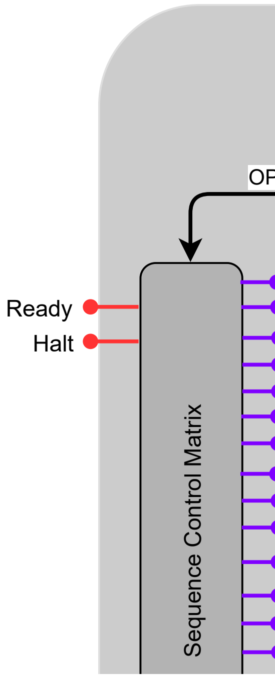


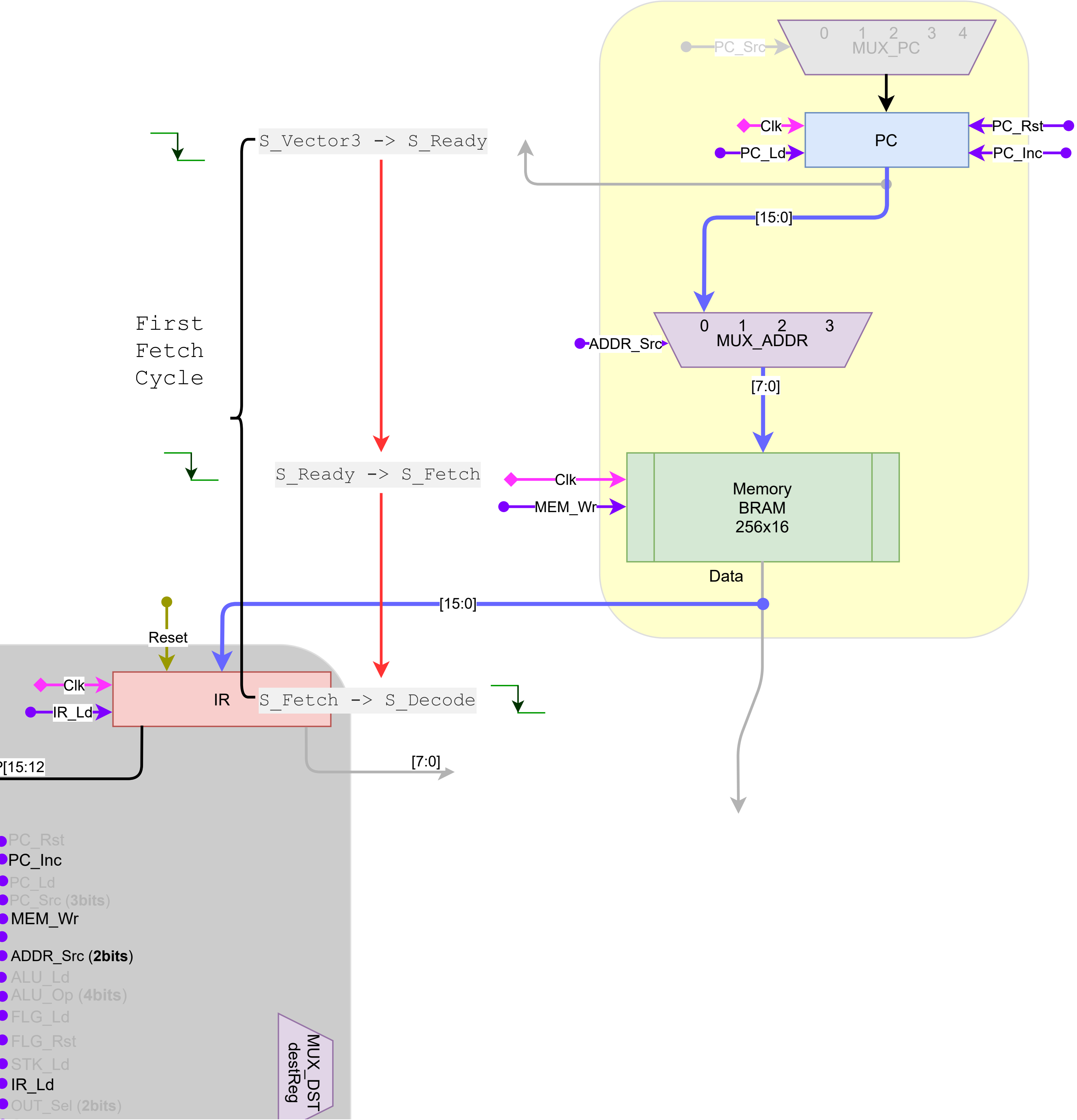


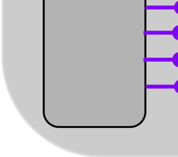

[15:0]
Return address
Reset
Branch Address
Add-1
Clk
Stack
1x16
STK_Ld
0 1 2 3 4
MUX_PC
PC_Src
Clk
PC
PC_Rst
PC_Ld
PC_Inc
Reset Vector = 0xFF
[15:0]
[15:0]
BRA_Src
MUX_BRA
0
1
[15:0]
0 1 2 3
MUX_ADDR
ADDR_Src
[7:0]
Sign extend
[7:0]
Clk
MEM_Wr
Memory
BRAM
256x16
Data
[15:0]
[15:0]
Reset
Clk
IR
IR_Ld
[10:3]
Zero extend H
[15:0]
[15:0]
[15:12
[7:0]
Zero extend L
LDI
0 1 2 3
MUX_DATA
DATA_Src
PC_Rst
PC_Inc
PC_Ld
PC_Src (3bits)
MEM_Wr
ADDR_Src (2bits)
ALU_Ld
ALU_Op (4bits)
FLG_Ld
FLG_Rst
STK_Ld
IR_Ld
OUT_Sel (2bits)
OUT_Ld
BRA_Src (1bit)
REG_WE
DATA_Src (2bits)
Reg Src 1 [2:0]
Reg Src 2 [5:3]
Register file
8x16
Clk
REG_WE
MUX_DST
destReg
Source 1
Source 2
Register Dest [8:6] or [10:8]
Instr
A
B
ALU_Op
ALU
Execution
Unit
[15:0]
Flags
Clk
ALU Flags
FLG_Rst
FLG_Ld
Reset
ALU Results
Clk
ALU_Ld

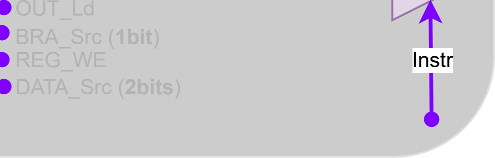

Source 1
0
1
MUX_OUT
Out_Sel
Reset
Output
Clk
Out_Ld
[15:0]
FPGA Pins


Reset Active
Reset
Vector 1
Vector 2
Vector 3
Halt
HALT
Decode
ALU
Execute
NOP,OTR
Execute
Fetch
Ready


OP
Ready
Halt
Sequence Control Matrix


First Fetch Cycle
S_Vector3 -> S_Ready
S_Ready -> S_Fetch
S_Fetch -> S_Decode
0 1 2 3 4
MUX_PC
PC_Src
Clk
PC_Ld
PC
PC_Rst
PC_Inc
[15:0]
0 1 2 3
MUX_ADDR
ADDR_Src
[7:0]
Clk
MEM_Wr
Memory
BRAM
256x16
Data
[15:0]
Reset
Clk
IR_Ld
IR
[15:12
[7:0]
PC_Rst
PC_Inc
PC_Ld
PC_Src (3bits)
MEM_Wr
ADDR_Src (2bits)
ALU_Ld
ALU_Op (4bits)
FLG_Ld
FLG_Rst
STK_Ld
IR_Ld
OUT_Sel (2bits)
MUX_DST
destReg

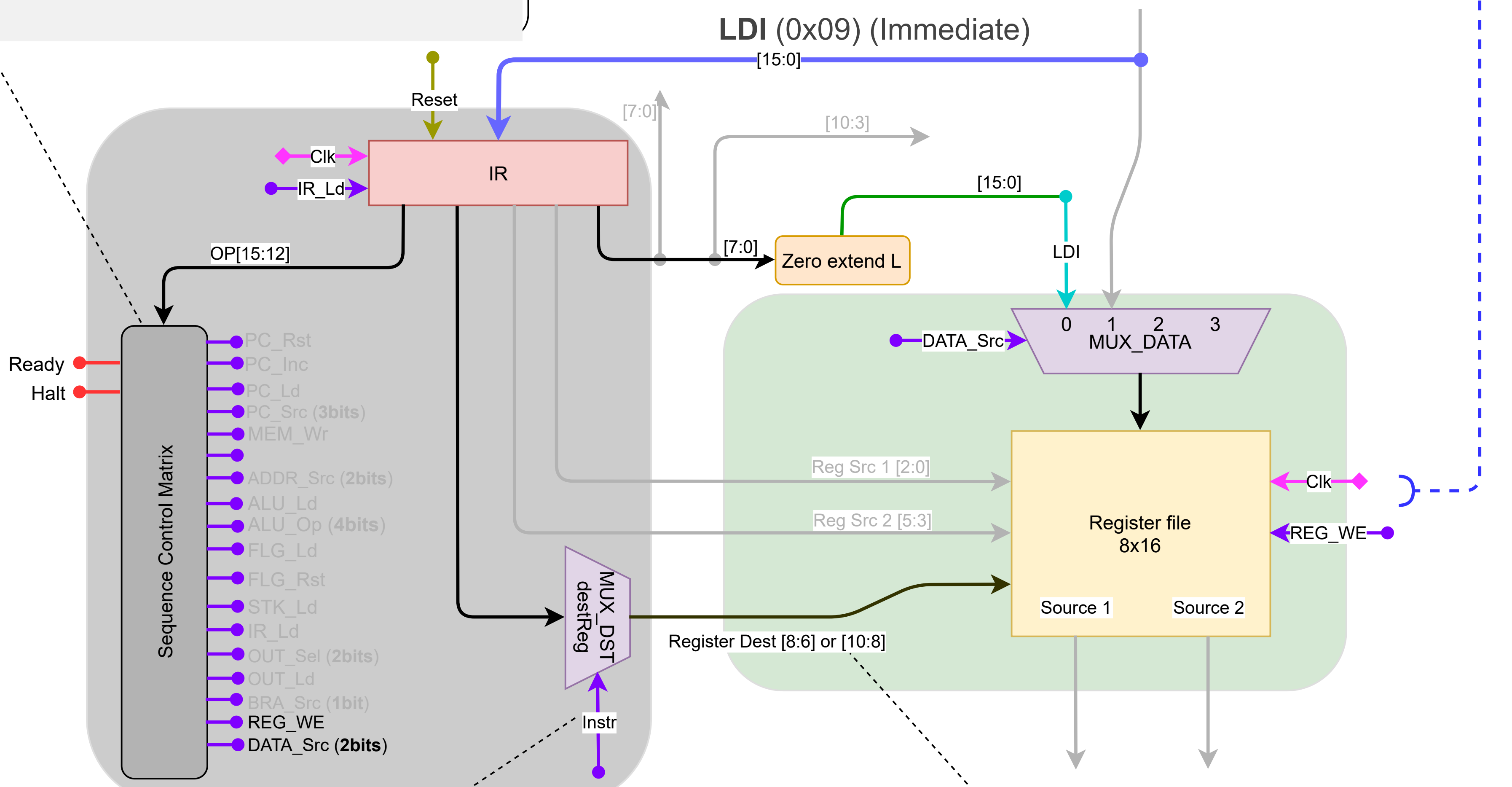



OUT_Ld
BRA_Src (1bit)
REG_WE
DATA_Src (2bits)
Instr

```
`LDI: begin // Load Immediate.
   reg_we = 1'b0; // Enable write to reg file
   data_src = 2'b00; // Select Zero extended-L source
end
```

S_Decode -> S_Execute

LDI (0x09) (Immediate)



LDI (0x09)

Dst ← #(V7:V0)

V(n) is loaded into the destination register Dst(n).

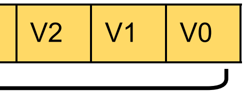
1	0	0	1		Dst	Dst	Dst	V7	V6	V5	V4	V3

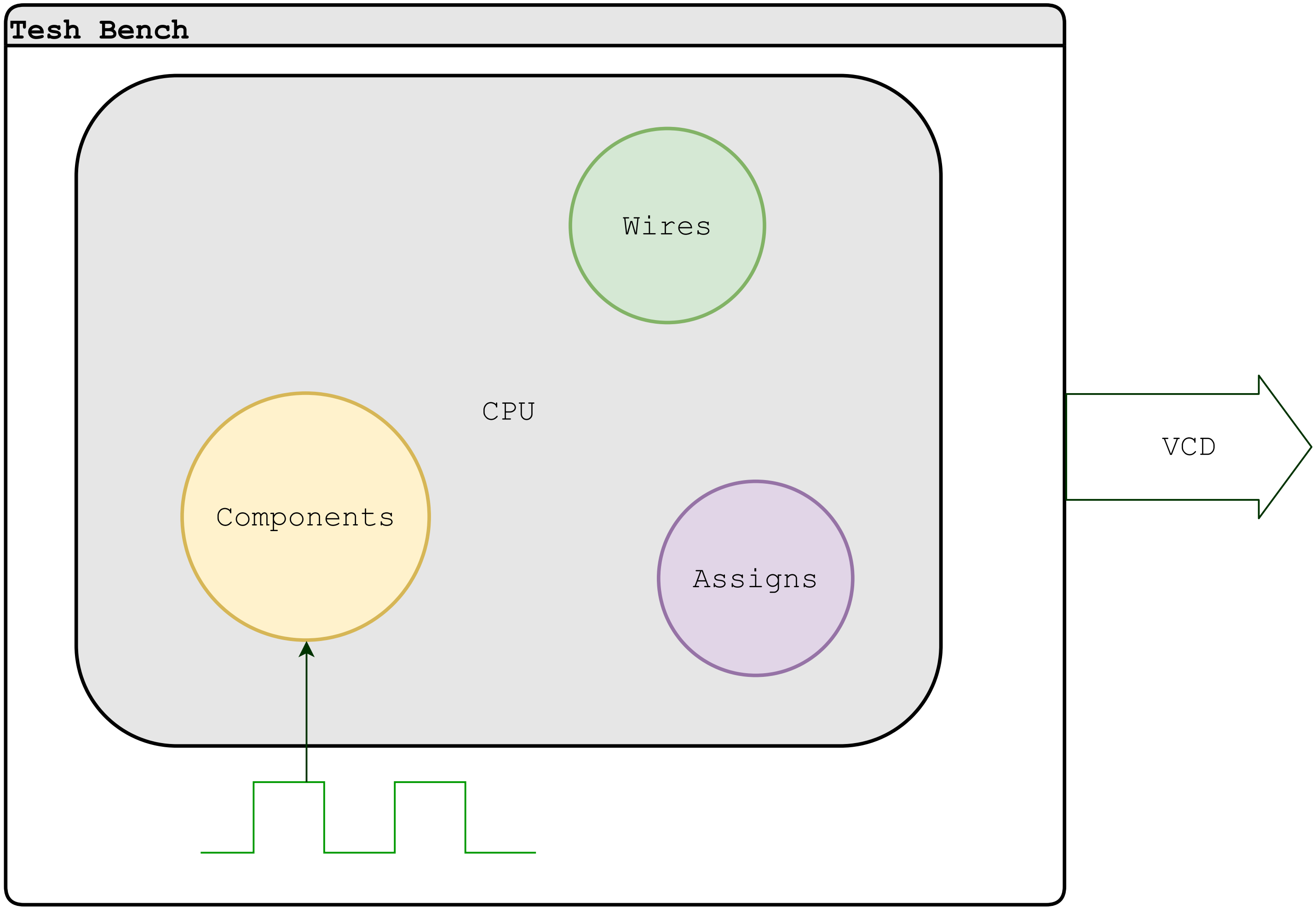
ir[10:8]   value

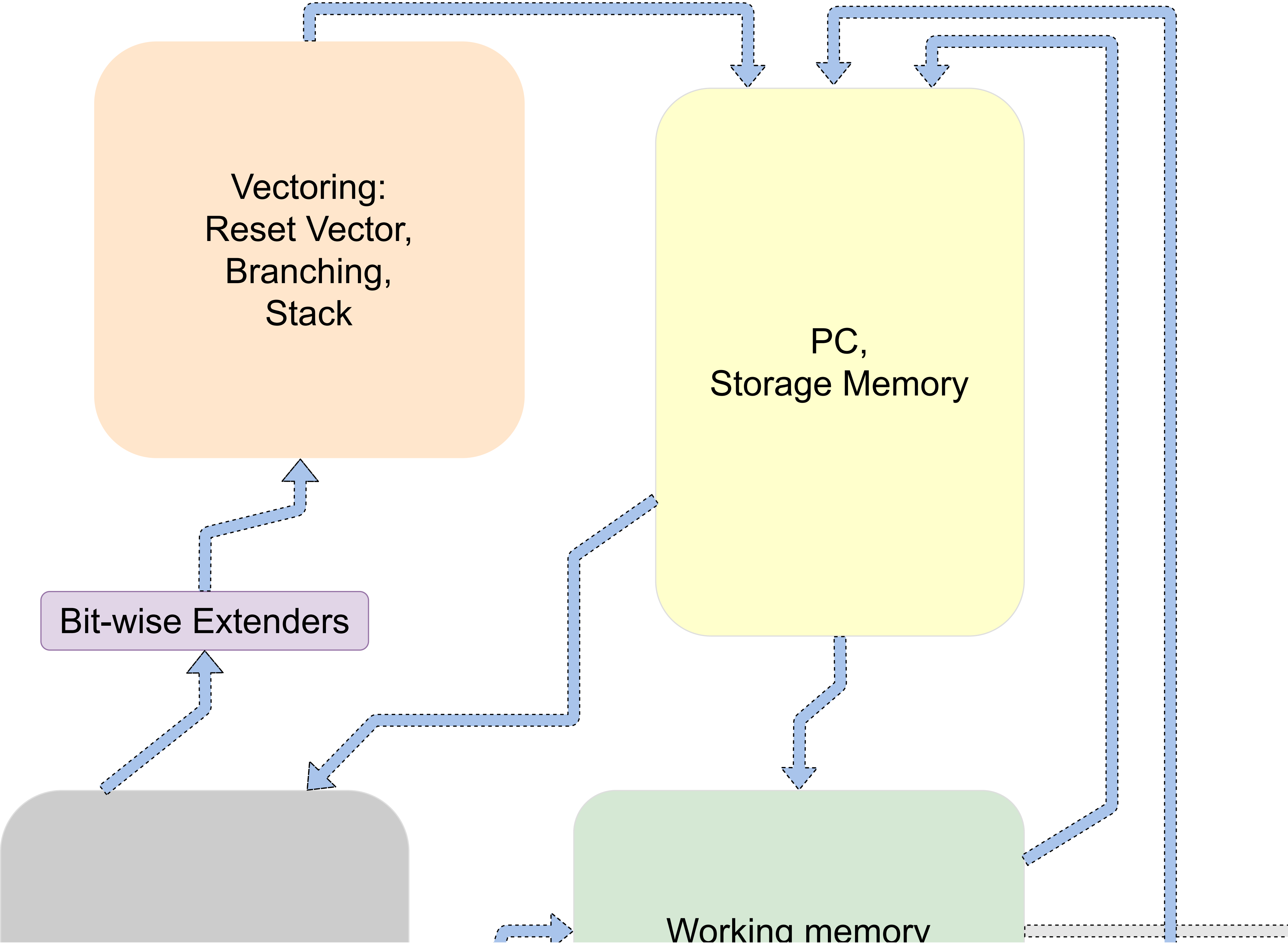
```
`define Instr ir[15:12]
`define DestRegLDI ir[10:8] // For LDI instruction

wire [2:0] destReg;
assign destReg = `Instr == `LDI ? `DestRegLDI :
`DestReg;
```

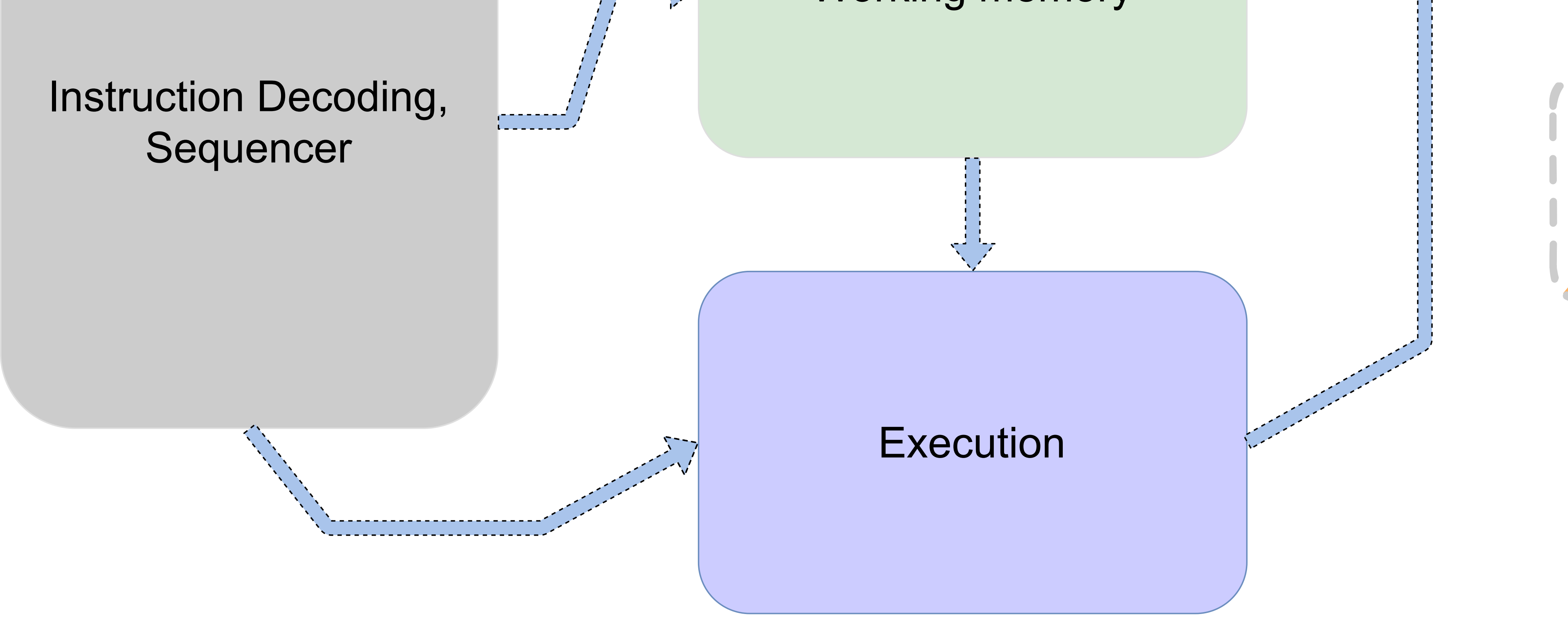
2 Input Mux

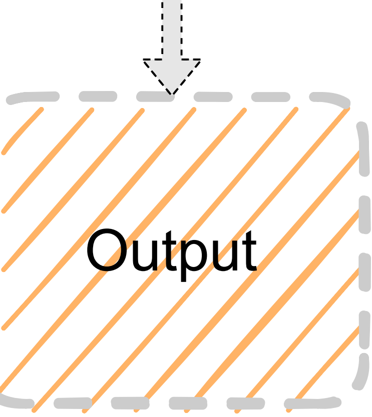

V2
V1
V0


Tesh Bench
Wires
CPU
VCD
Components
Assigns

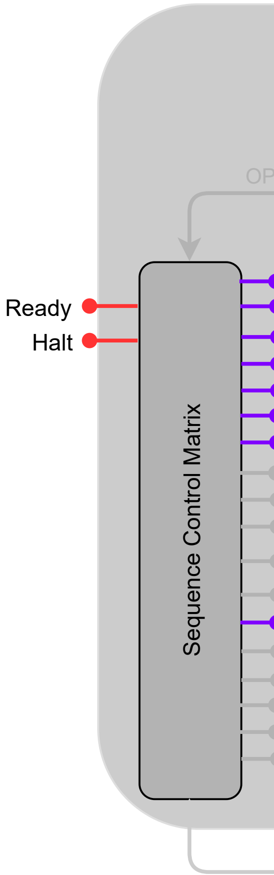

Vectoring:
Reset Vector,
Branching,
Stack
PC,
Storage Memory
Bit-wise Extenders
Working memory

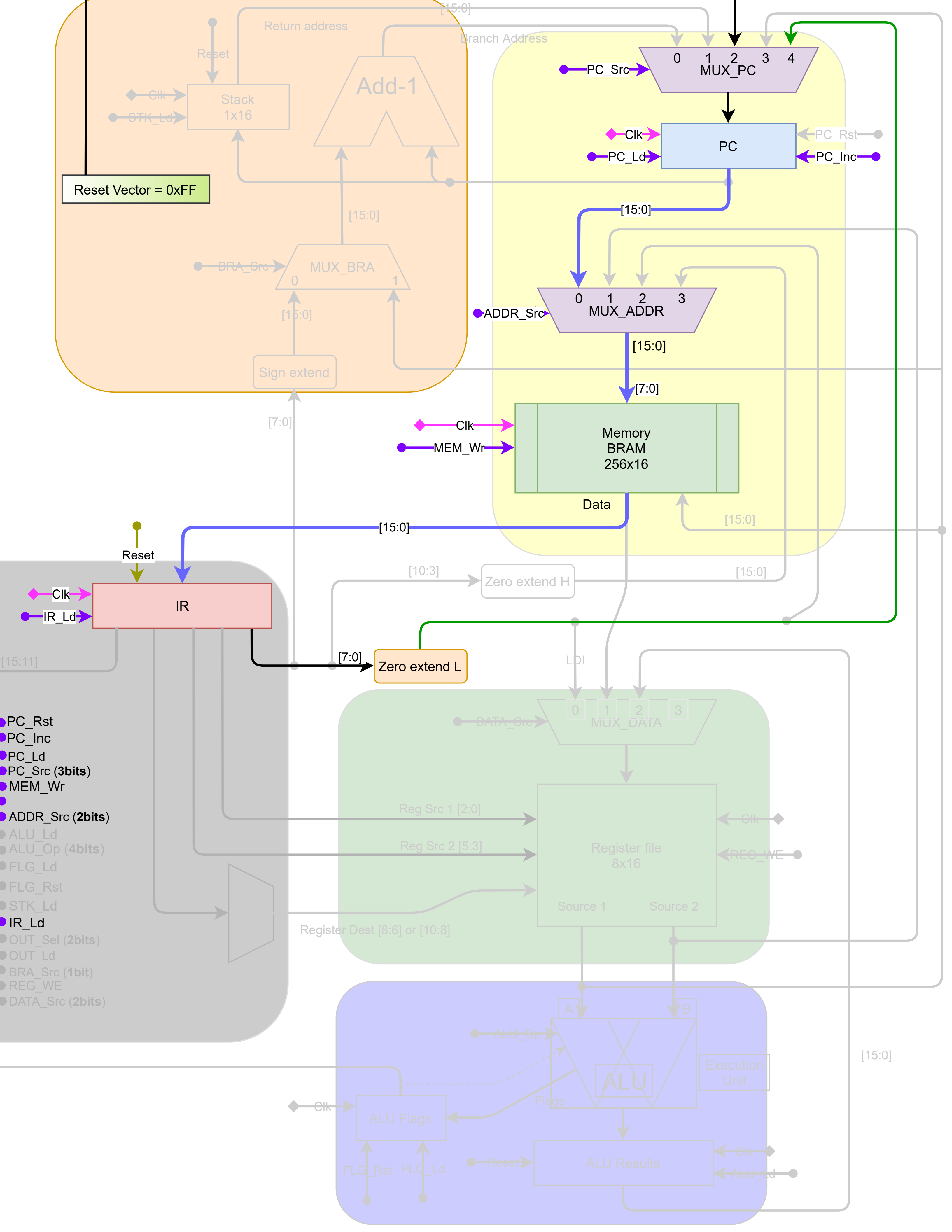
```
------- Top MEM contents ------
memory[  0] = 0001000000010000 <- 1010
memory[  1] = 0000000100000001 <- 0101
memory[  2] = 0001000000010000 <- 1010
memory[  3] = 0000000100000001 <- 0101
memory[  4] = 1111111100000000 <- ff00
memory[  5] = 0001000000000000 <- 1000
memory[  6] = 1011000000000000 <- b000
memory[  7] = xxxxxxxxxxxxxxxx <- xxxx
memory[  8] = xxxxxxxxxxxxxxxx <- xxxx
memory[  9] = xxxxxxxxxxxxxxxx <- xxxx
memory[ 10] = xxxxxxxxxxxxxxxx <- xxxx
memory[ 11] = xxxxxxxxxxxxxxxx <- xxxx
memory[ 12] = xxxxxxxxxxxxxxxx <- xxxx
memory[ 13] = xxxxxxxxxxxxxxxx <- xxxx
memory[ 14] = xxxxxxxxxxxxxxxx <- xxxx
------- Bottom MEM contents ------
memory[250] = xxxxxxxxxxxxxxxx <- xxxx
memory[251] = xxxxxxxxxxxxxxxx <- xxxx
memory[252] = xxxxxxxxxxxxxxxx <- xxxx
memory[253] = xxxxxxxxxxxxxxxx <- xxxx
memory[254] = xxxxxxxxxxxxxxxx <- xxxx
memory[255] = 0000000000000101 <- 0005
```


Instruction Decoding, Sequencer
Working memory
Execution

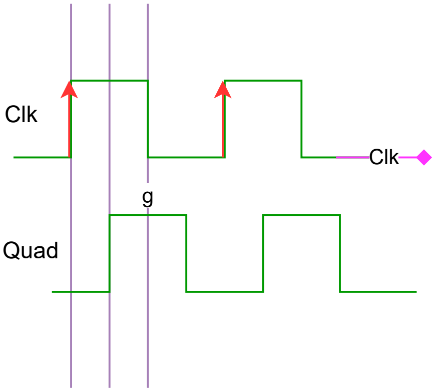

Output

000 001 010 011 100

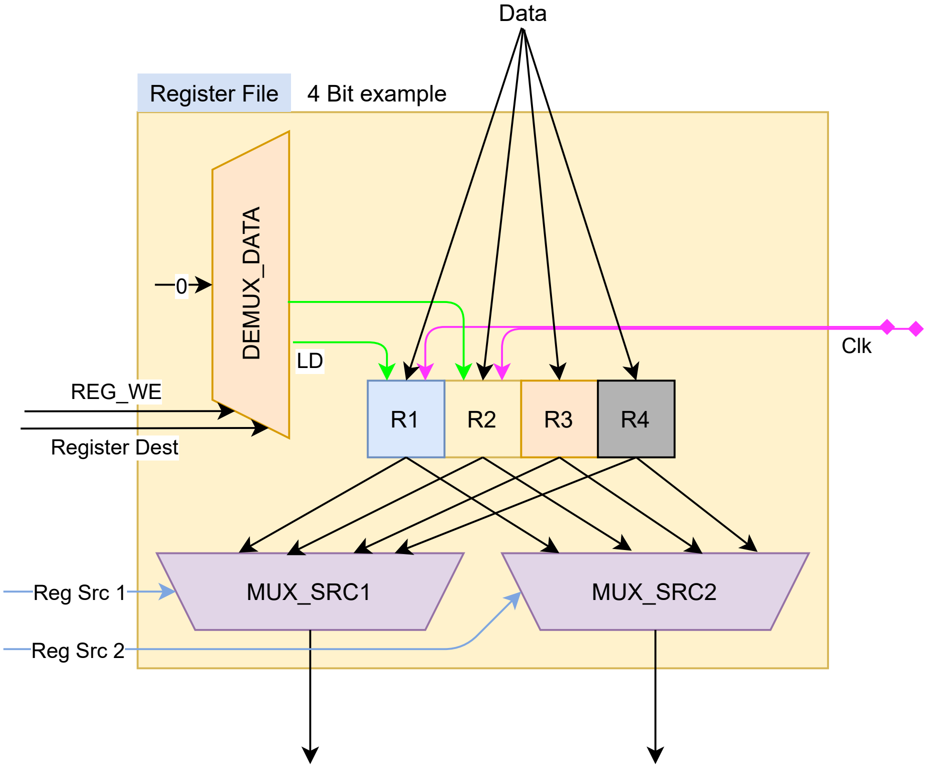

OP
Ready
Halt
Sequence Control Matrix



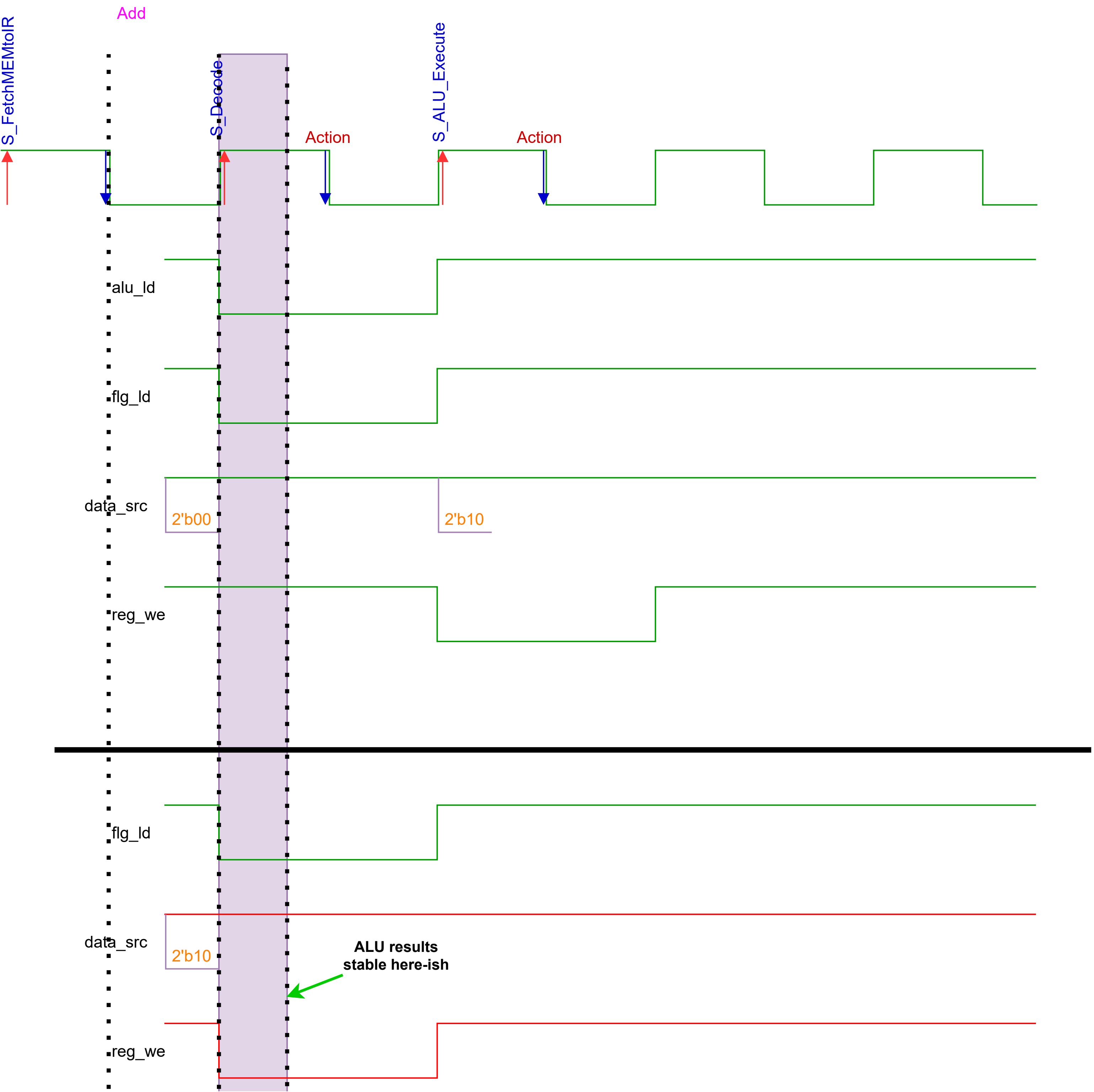
[15:0]
Return address
Branch Address
Reset
Clk
STK_Ld
Stack
1x16
Add-1
Reset Vector = 0xFF
[15:0]
BRA_Src
MUX_BRA
0
1
[15:0]
Sign extend
[7:0]
0
1
2
3
4
MUX_PC
PC_Src
Clk
PC_Ld
PC
PC_Rst
PC_Inc
[15:0]
0
1
2
3
MUX_ADDR
ADDR_Src
[15:0]
[7:0]
Clk
MEM_Wr
Memory
BRAM
256x16
Data
[15:0]
[15:0]
Reset
Clk
IR_Ld
IR
[10:3]
Zero extend H
[15:0]
[15:11]
[7:0]
Zero extend L
LDI
PC_Rst
PC_Inc
PC_Ld
PC_Src (3bits)
MEM_Wr
ADDR_Src (2bits)
ALU_Ld
ALU_Op (4bits)
FLG_Ld
FLG_Rst
STK_Ld
IR_Ld
OUT_Sel (2bits)
OUT_Ld
BRA_Src (1bit)
REG_WE
DATA_Src (2bits)
DATA_Src
0
1
2
3
MUX_DATA
Reg Src 1 [2:0]
Reg Src 2 [5:3]
Register file
8x16
Clk
REG_WE
Source 1
Source 2
Register Dest [8:6] or [10:8]
A
B
ALU_Op
ALU
Execution
Unit
Flags
[15:0]
Clk
ALU Flags
FLG_Rst
FLG_Ld
Reset
ALU Results
Clk
ALU_Ld

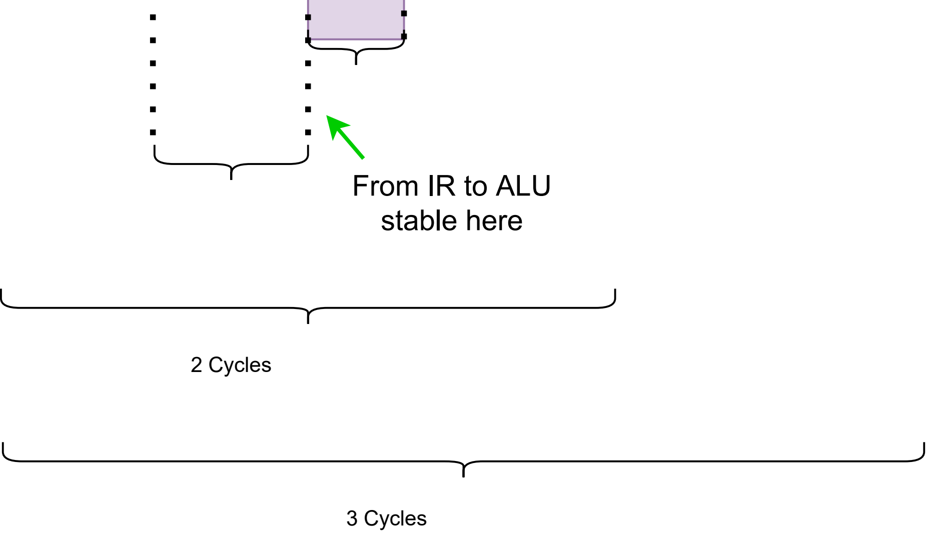


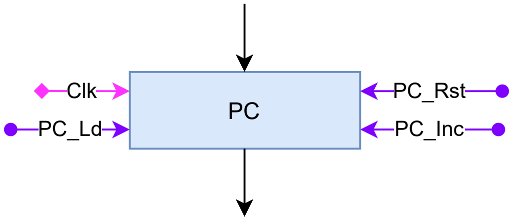
Instruction

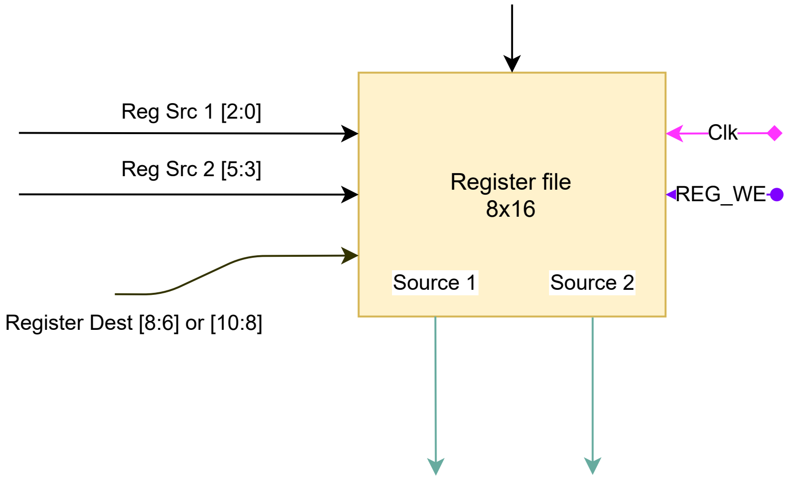


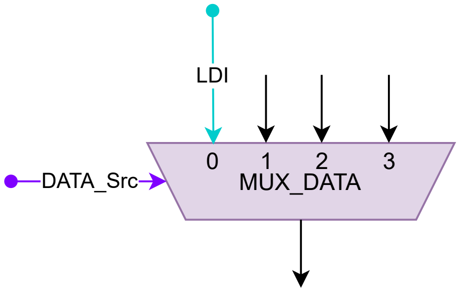
Clock

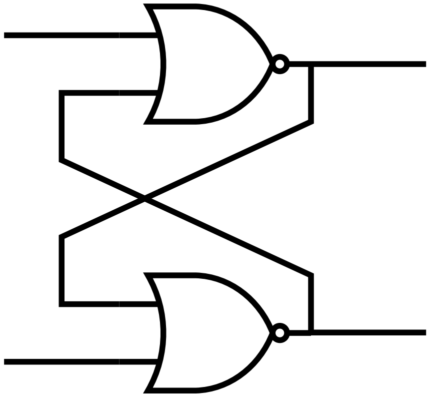

Add
S_FetchMEMtoIR
S_Decode
S_ALU_Execute
Action
Action
alu_ld
flg_ld
data_src
2'b00
2'b10
reg_we
flg_ld
data_src
2'b10
reg_we
ALU results
stable here-ish

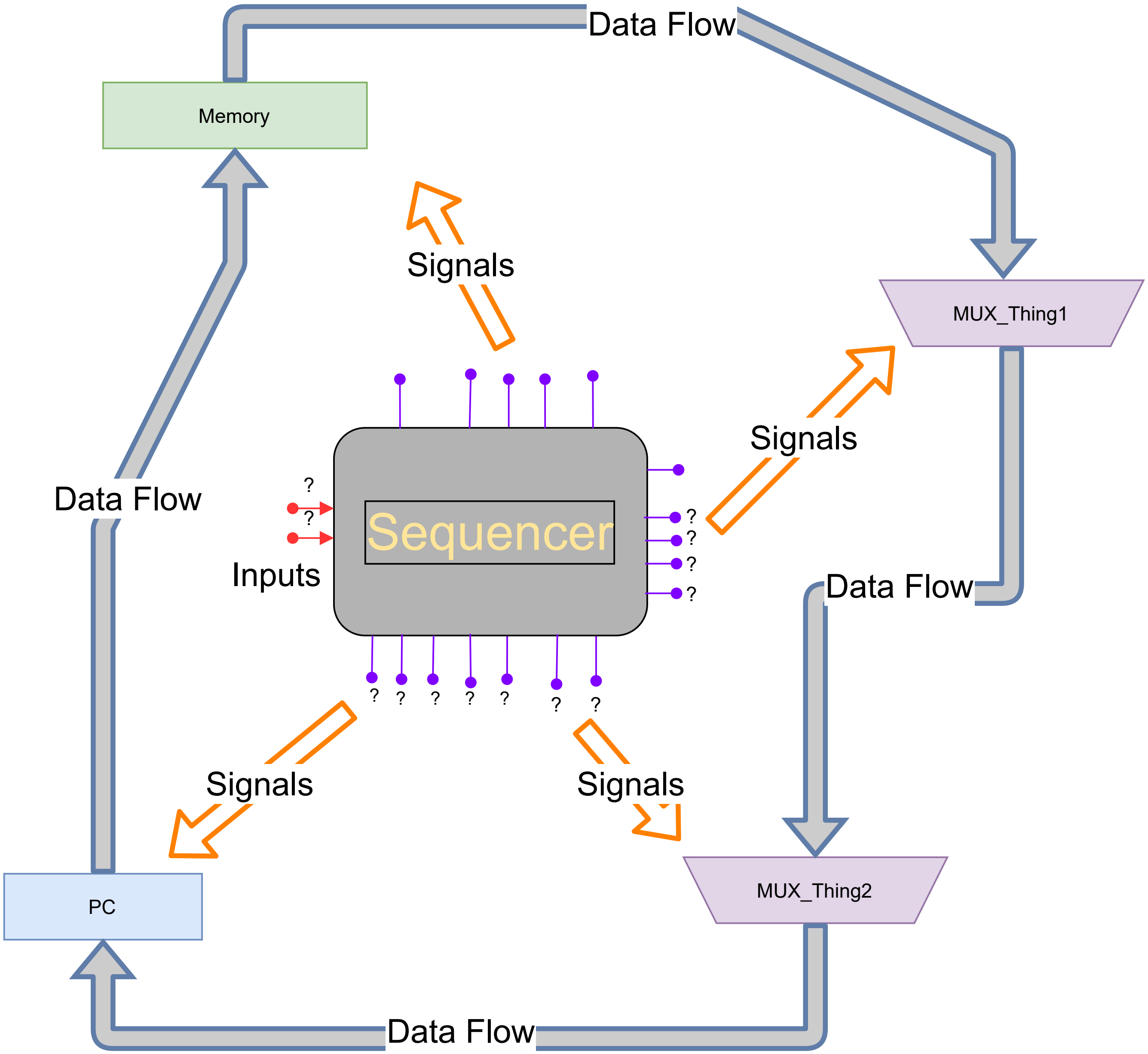

From IR to ALU stable here
2 Cycles
3 Cycles

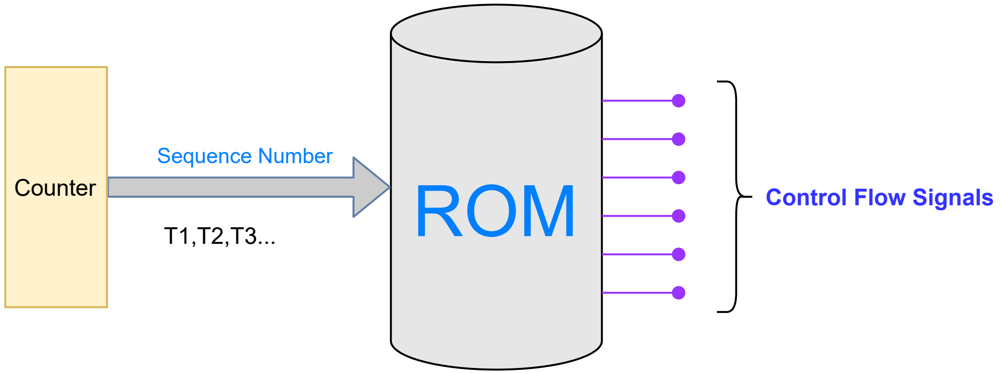

Clk
PC_Ld
PC
PC_Rst
PC_Inc

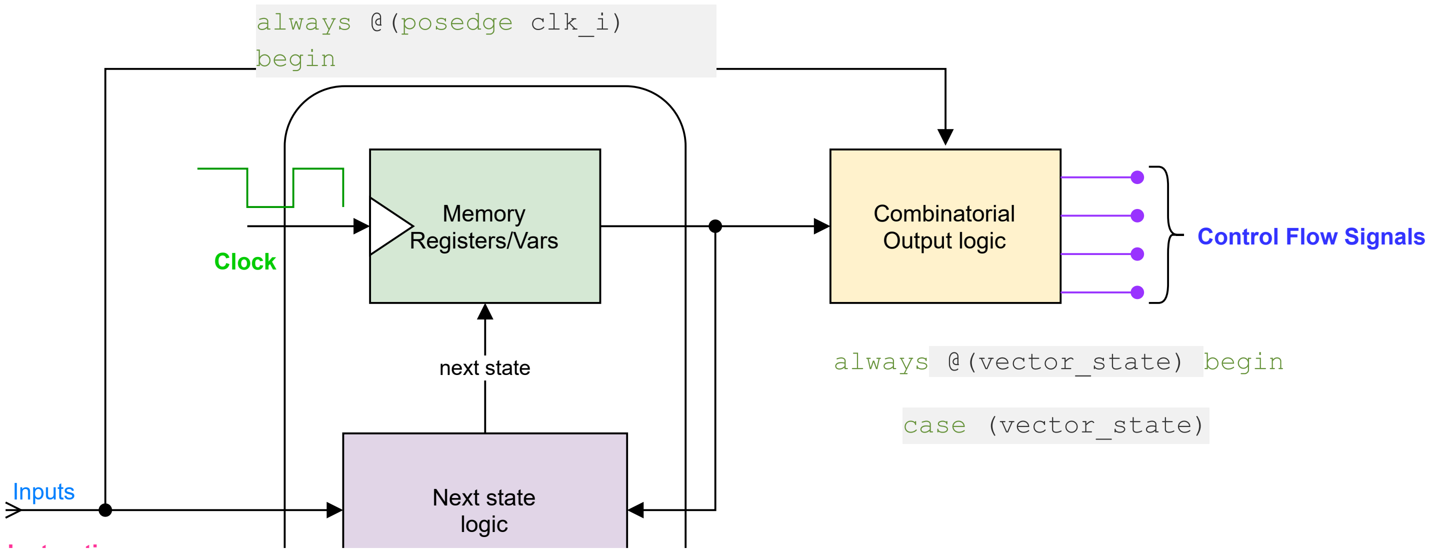

Reg Src 1 [2:0]
Reg Src 2 [5:3]
Register Dest [8:6] or [10:8]
Register file
8x16
Source 1
Source 2
Clk
REG_WE


LDI
0
1
2
3
MUX_DATA
DATA_Src




Data Flow
Memory
Signals
MUX_Thing1
Signals
Data Flow
?
?
Sequencer
Inputs
?
?
?
?
Data Flow
? ? ? ? ? ? ?
Signals
Signals
PC
MUX_Thing2
Data Flow

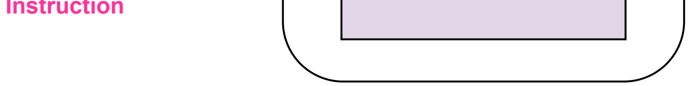


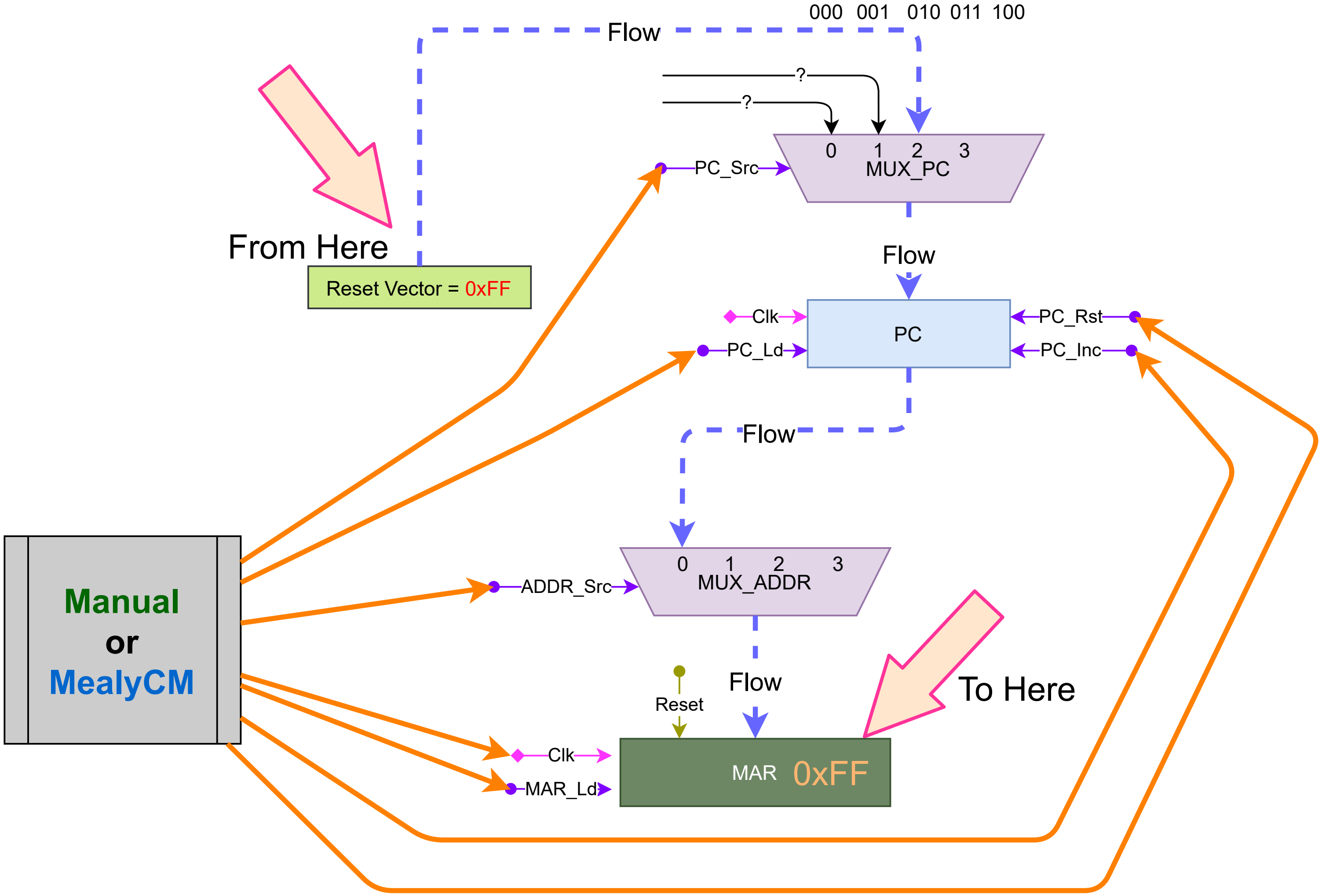


```
always @(posedge clk_i)
begin
```

```
always @(vector_state) begin
```

```
case (vector_state)
```


Instruction


000 001 010 011 100
Flow
?
?
0 1 2 3
MUX_PC
PC_Src
From Here
Reset Vector = 0xFF
Flow
PC
Clk
PC_Ld
PC_Rst
PC_Inc
Flow
Manual
or
MealyCM
0 1 2 3
MUX_ADDR
ADDR_Src
Flow
Reset
To Here
Clk
MAR_Ld
MAR 0xFF