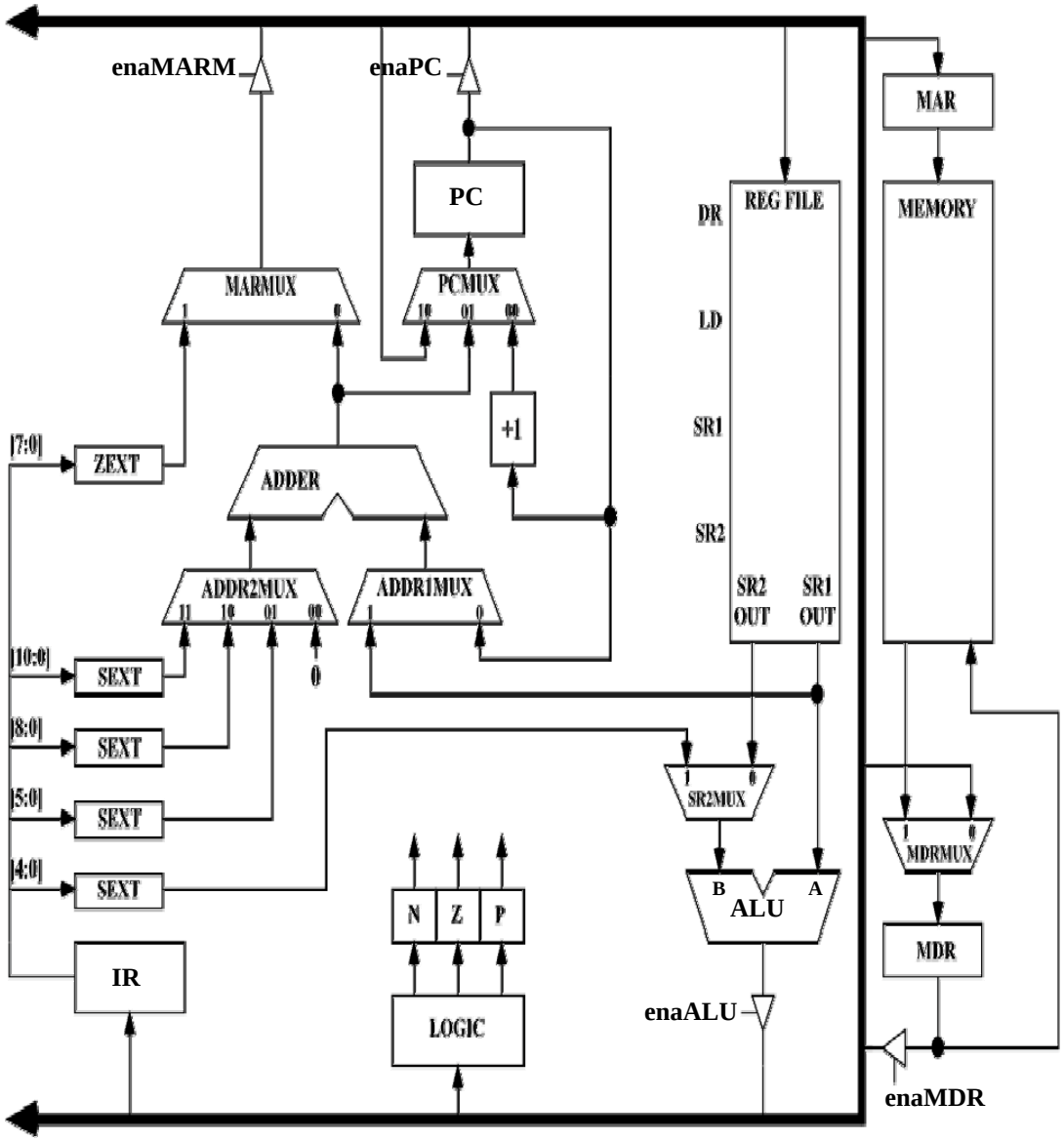
LC3-2
The LC-3 Datapath

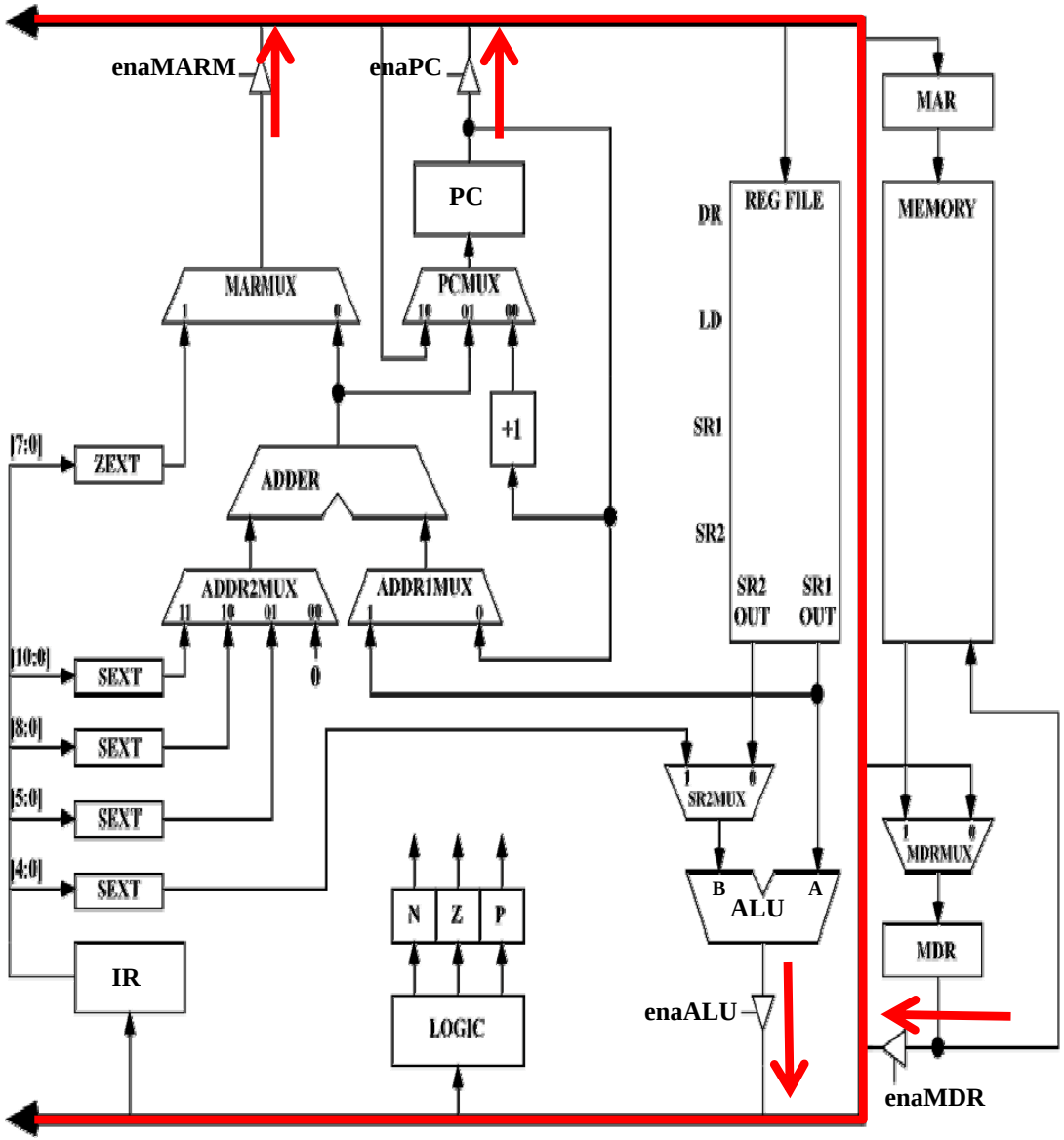


Datapath Strategy

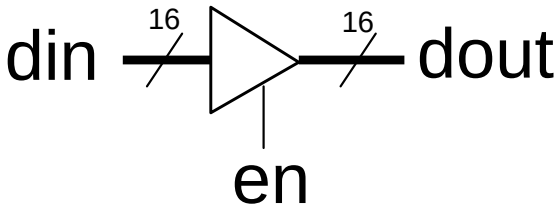
- Take datapath elements one-by-one
- Determine how each works
- Clearly identify inputs/outputs and needed control
- Design the element
- When all done, doing control is straightforward…
  - Push all the complexity into the datapath that we can

The Bus

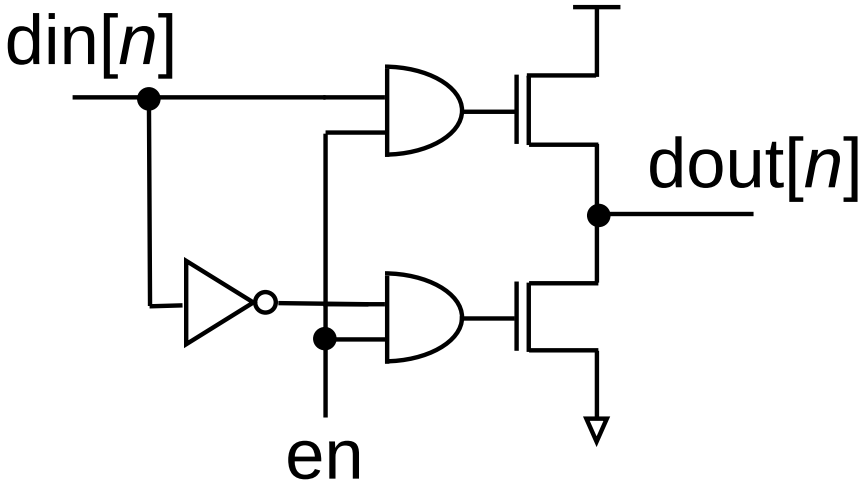
- Four major drivers: MARMUX, PC, ALU, MDR
- It is the means of data transfer between units.

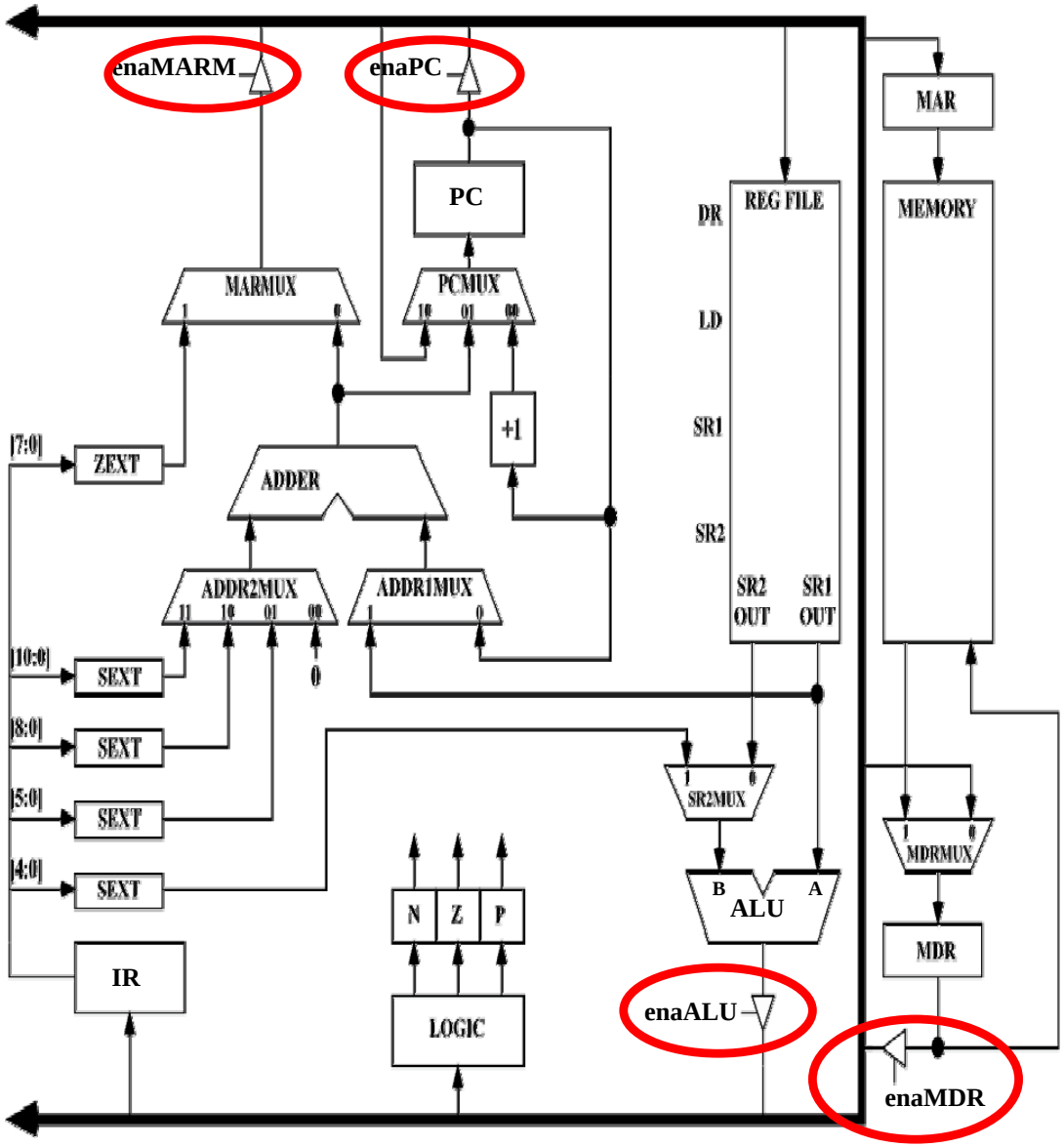


Tri-State Drivers



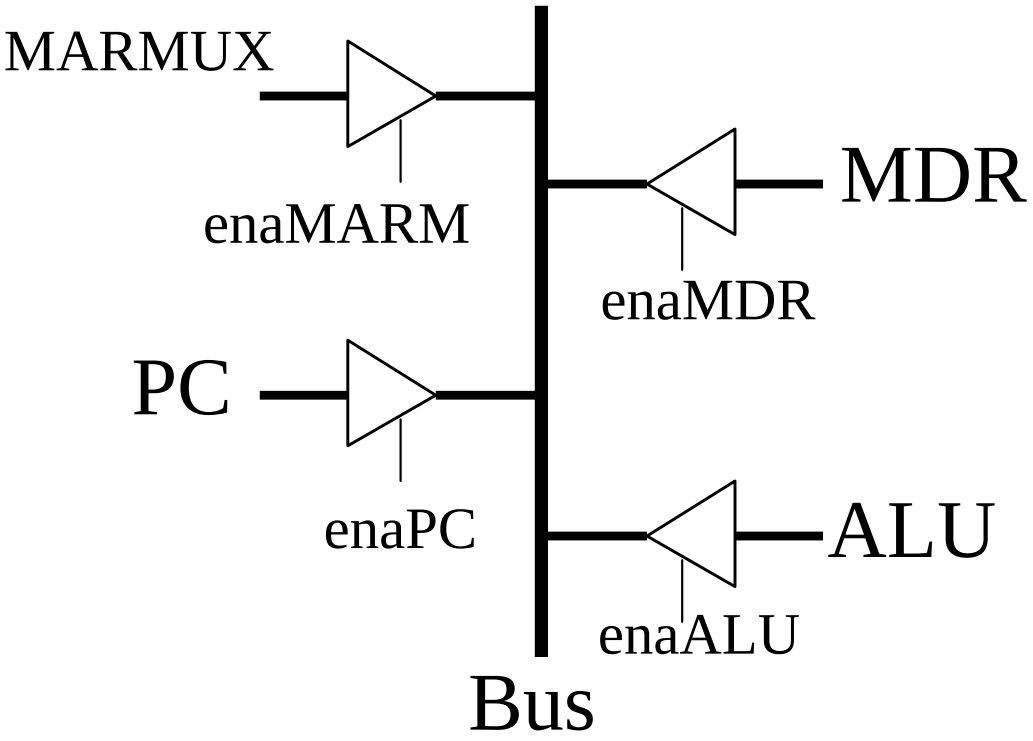

```
module ts_driver ( din, dout, en );
  input [15:0] din;
  output [15:0] dout;
  input en;
  assign dout = en ? din : 16'hZZZZ;
endmodule
```





More Tri-State

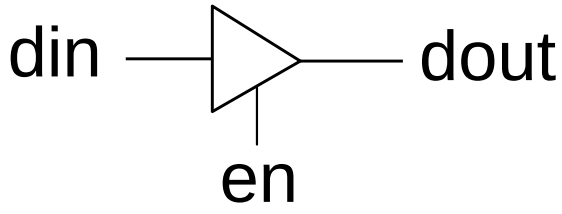
- Lots of drivers and lots of listeners
  - *Only one* driver on at any time
  - It's a distributed MUX!



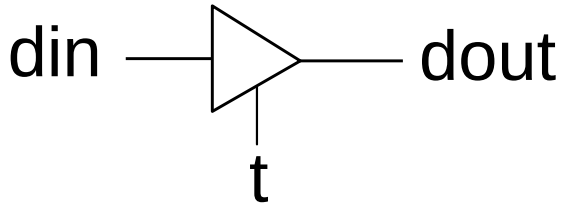
Be careful!
If more than one driver is on at once there will be contention on the bus with unpredictable results

Typical Tri-State Drivers

en	din	dout
0	0	Z
0	1	Z
1	0	0
1	1	1



t	din	dout
0	0	0
0	1	1
1	0	Z
1	1	Z

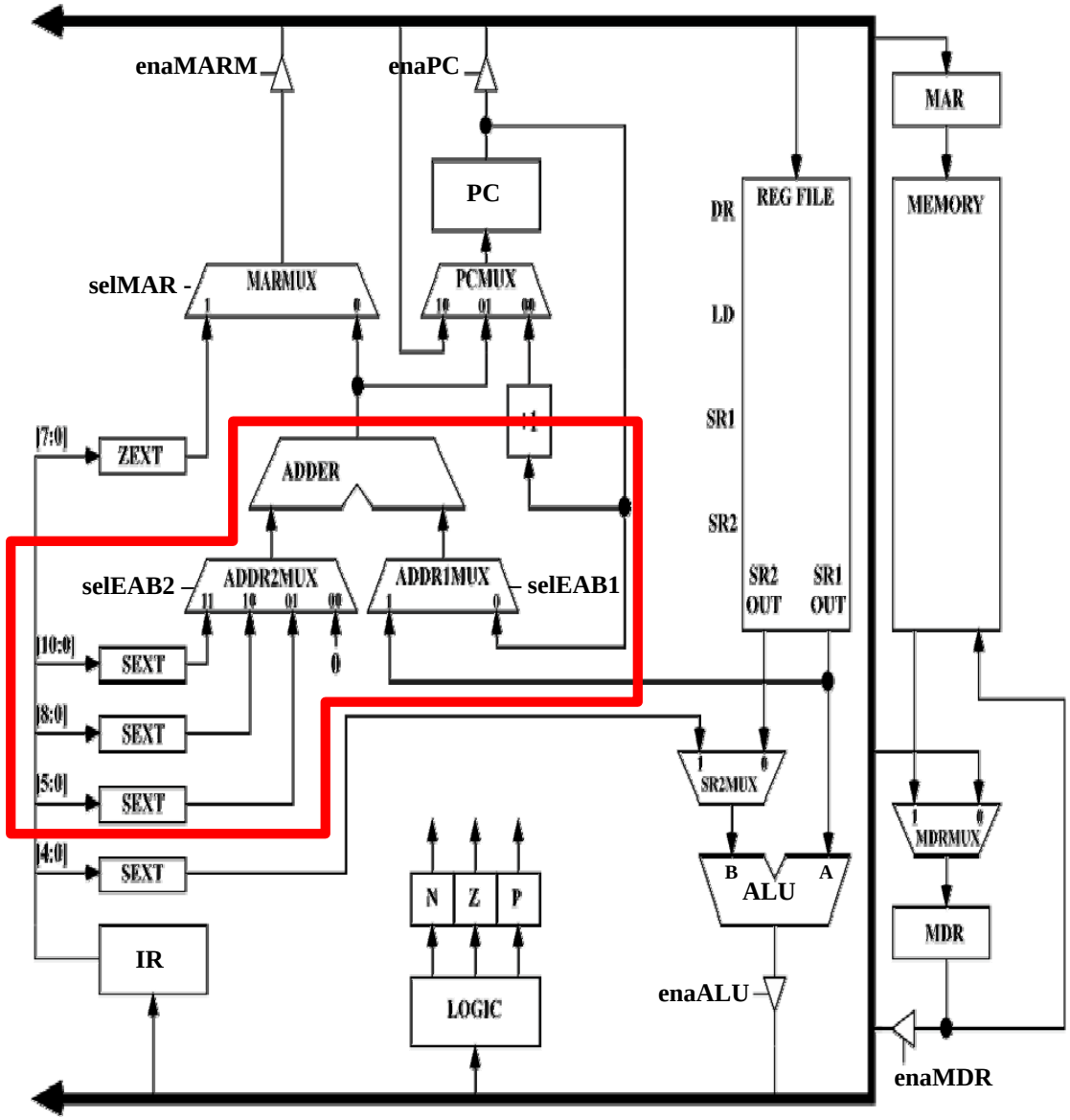


The Effective Address Block

- An adder, two muxes, and three sign extenders

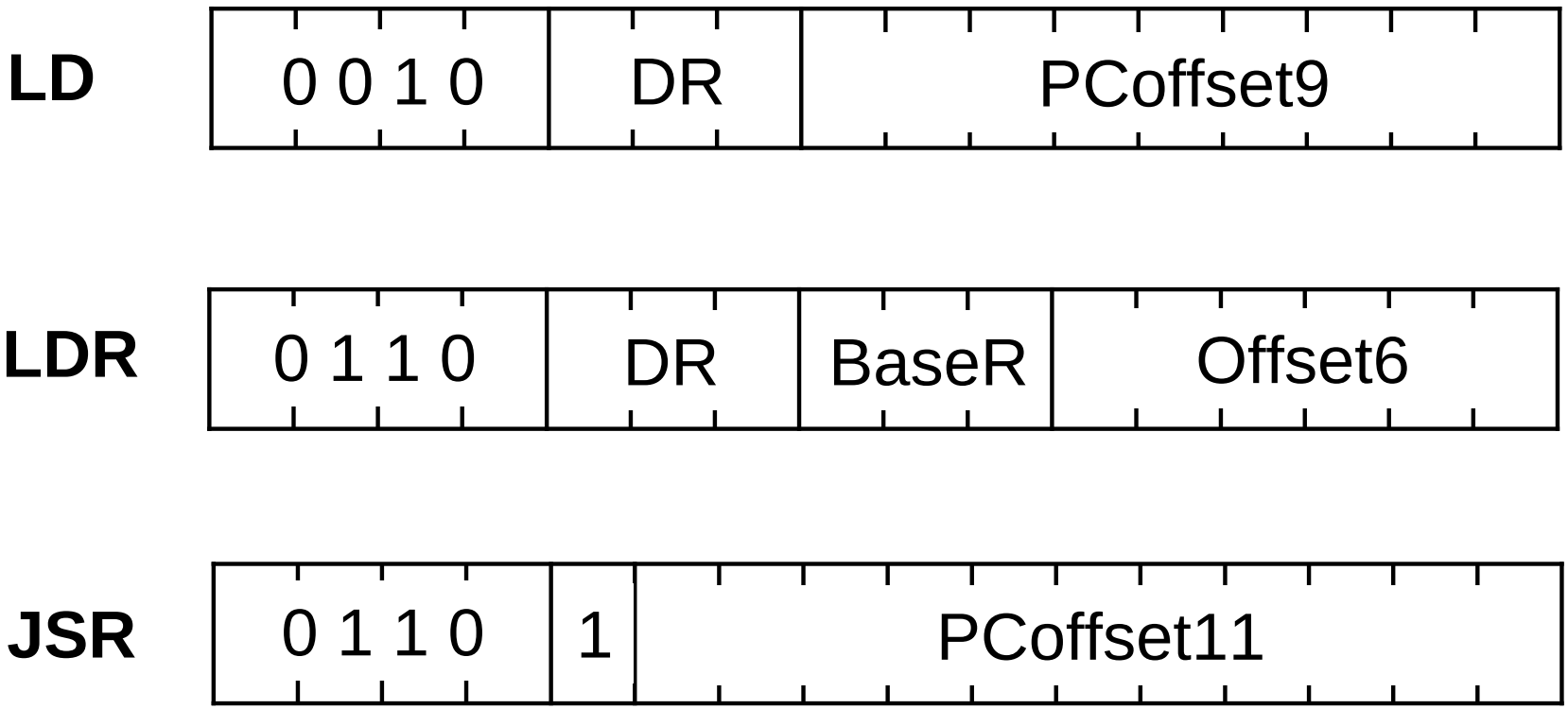
- Signals:

  IR[10:0]
  Ra[15:0]
  PC[15:0]
  selEAB1
  selEAB2[1:0]
  eabOut[15:0]

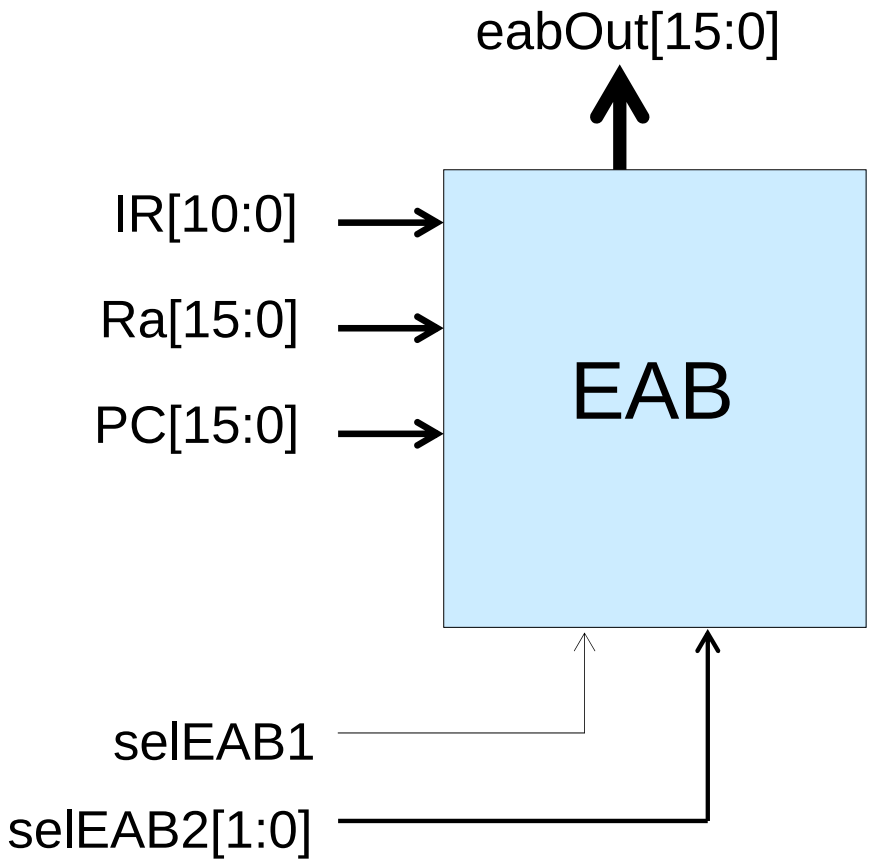


The Effective Address Block

Why are there 3 different sign extenders?



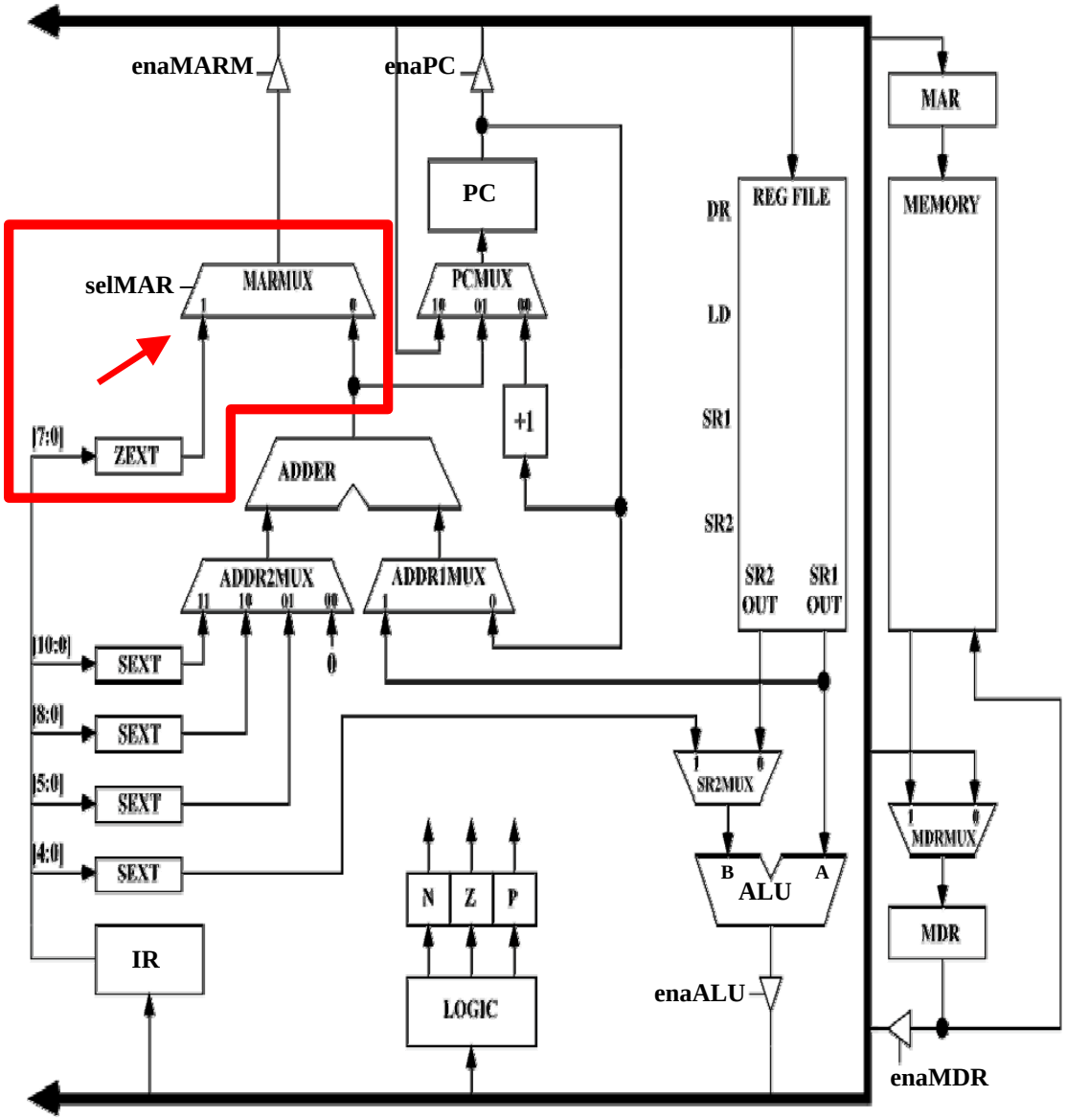
Designing the EAB



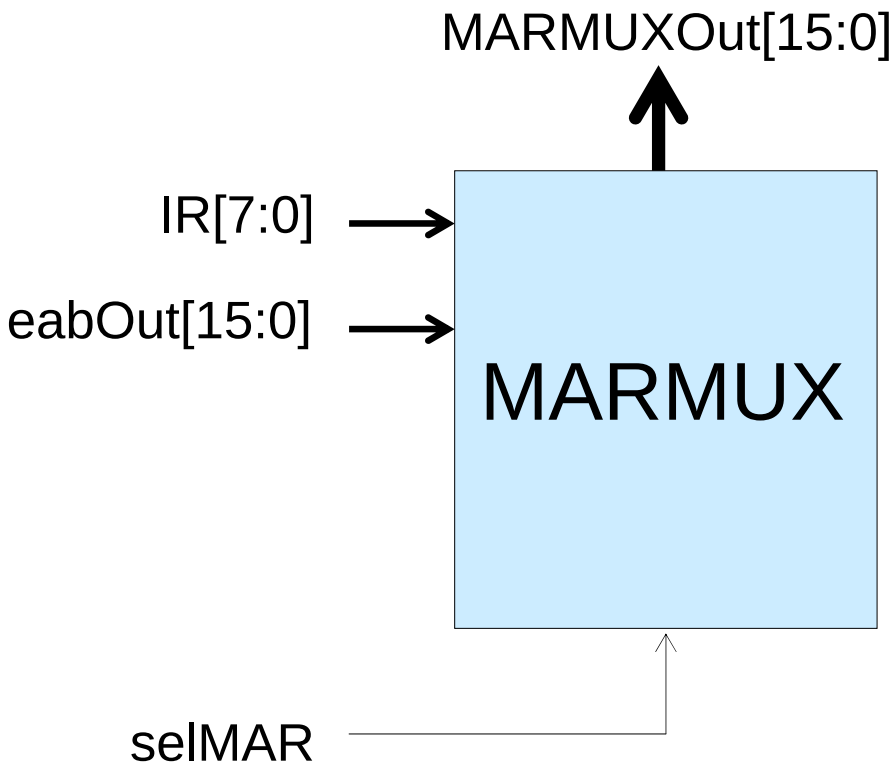
The MARMUX

- Simply a MUX and a zero extender
- Signals:

  IR[7:0]
  eabOut[15:0]
  selMAR
  MARMUXOut[15:0]



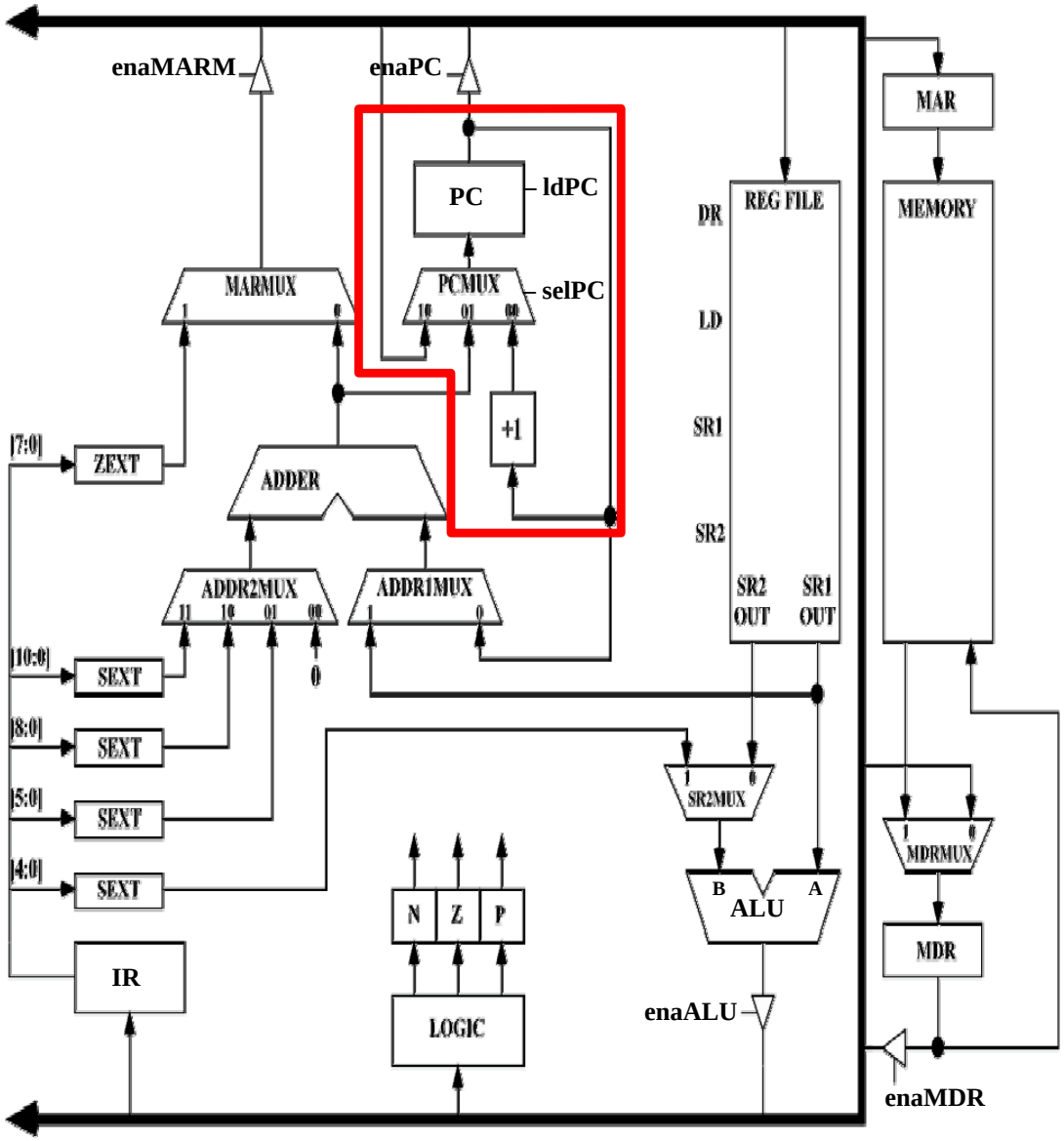
Designing the MARMUX



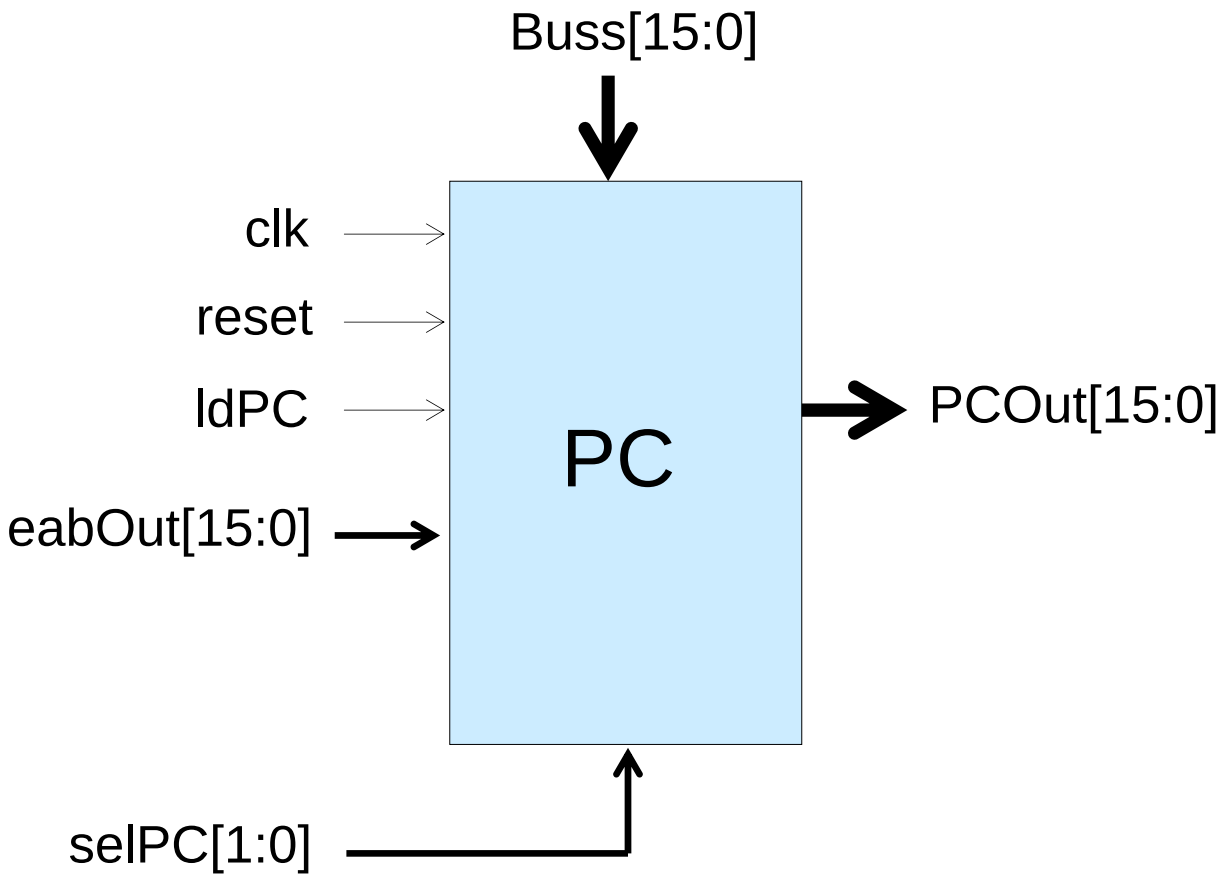
The PC

- A register, MUX, and an incrementer
- Signals:

  clk
  reset
  selPC[1:0]
  eabOut[15:0]
  ldPC
  PCOut[15:0]
  Buss[15:0]



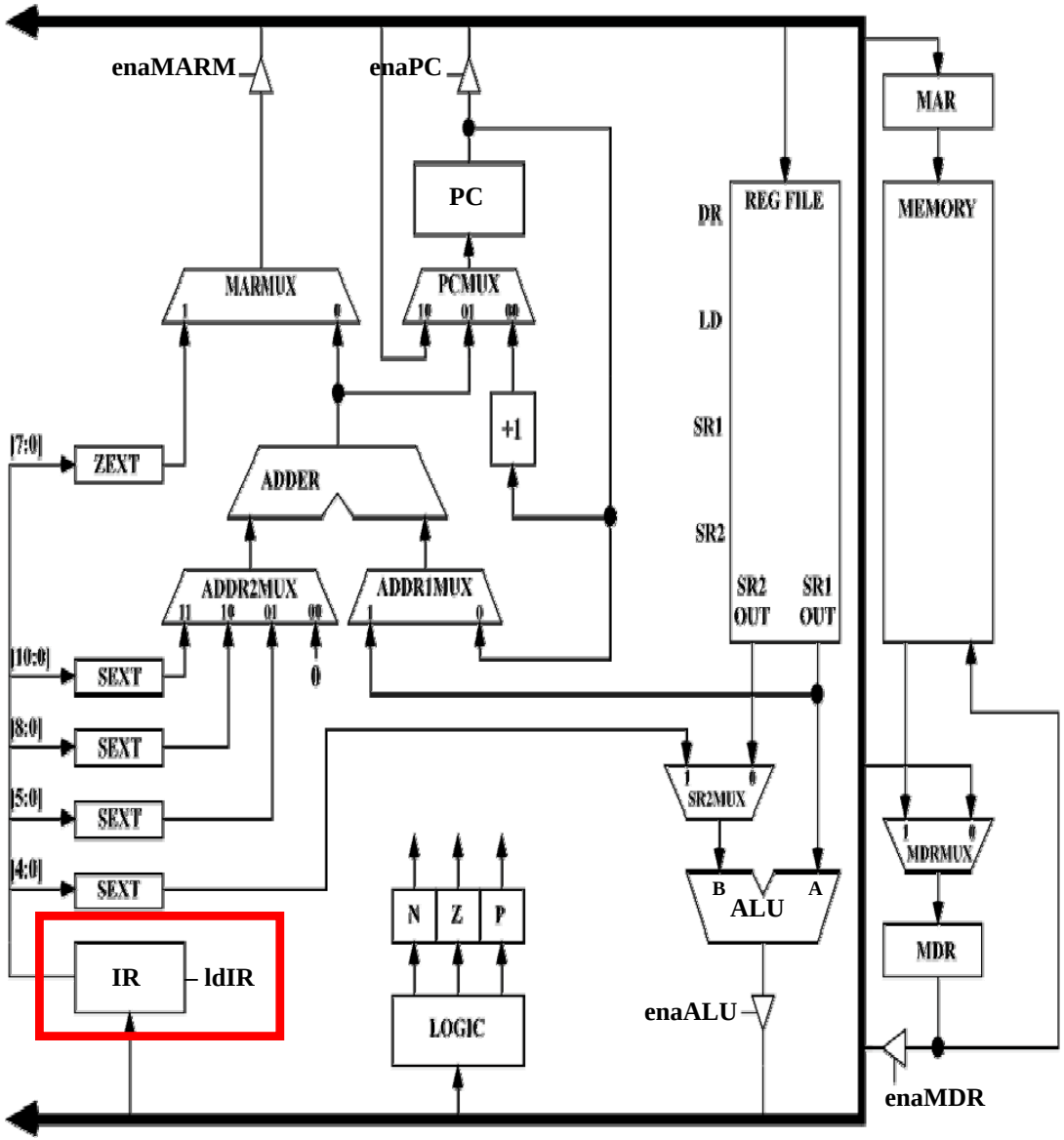
Designing the PC



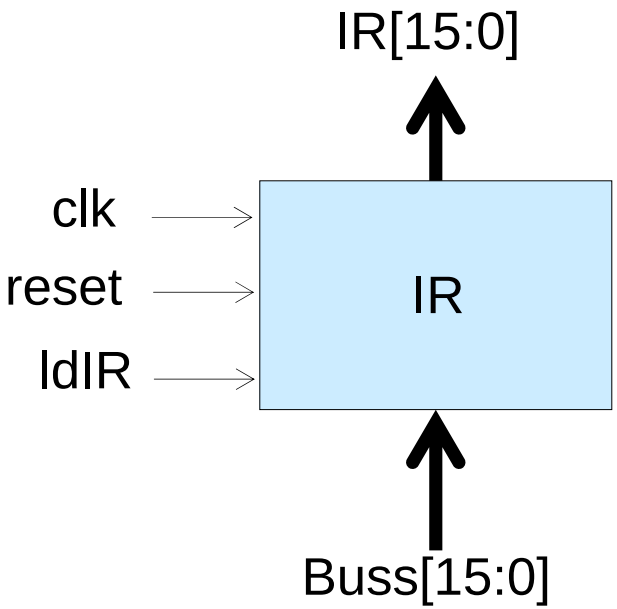
NOTE: Be sure your PC initializes to 0 on a global reset

The IR

- Simply a loadable register, loads from bus when enabled for loading
- Signals:
  - clk
  - reset
  - ldIR
  - Buss[15:0]
  - IR[15:0]



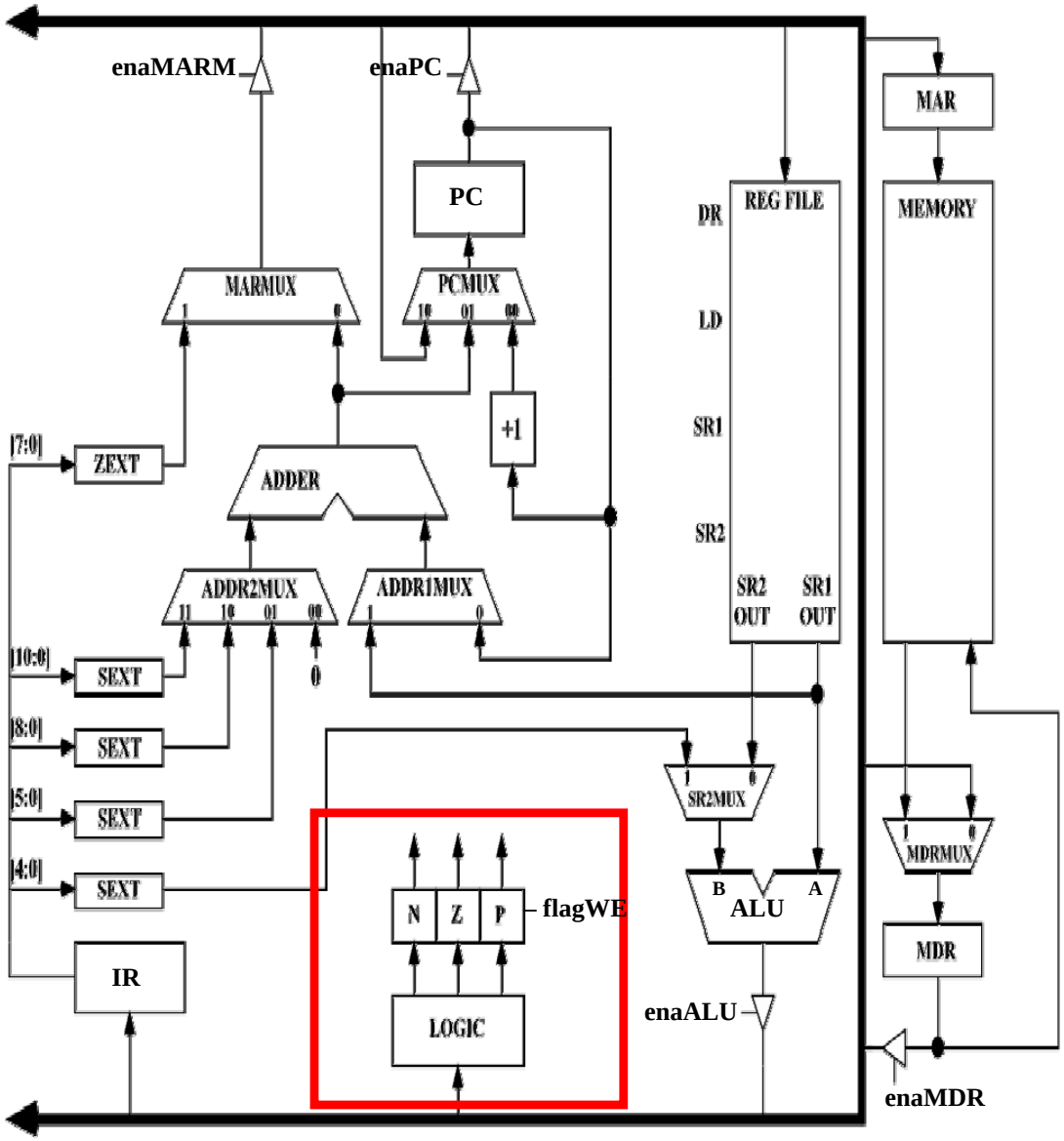
Designing the IR



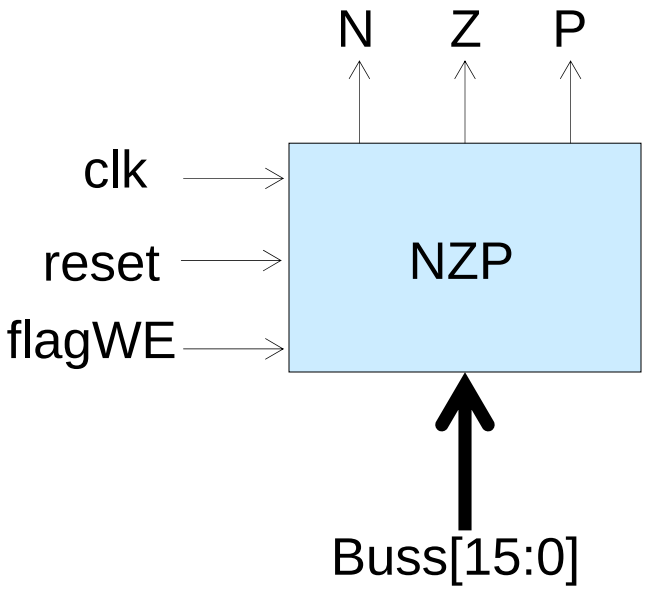
The nzp Flags

- Loadable registers which contain flags.
- Load on ***some*** reg writes
- Decode bus to determine flag values
- Signals:

  clk
  reset
  flagWE
  Buss[15:0]
  N
  Z
  P



Designing the nzp Flags

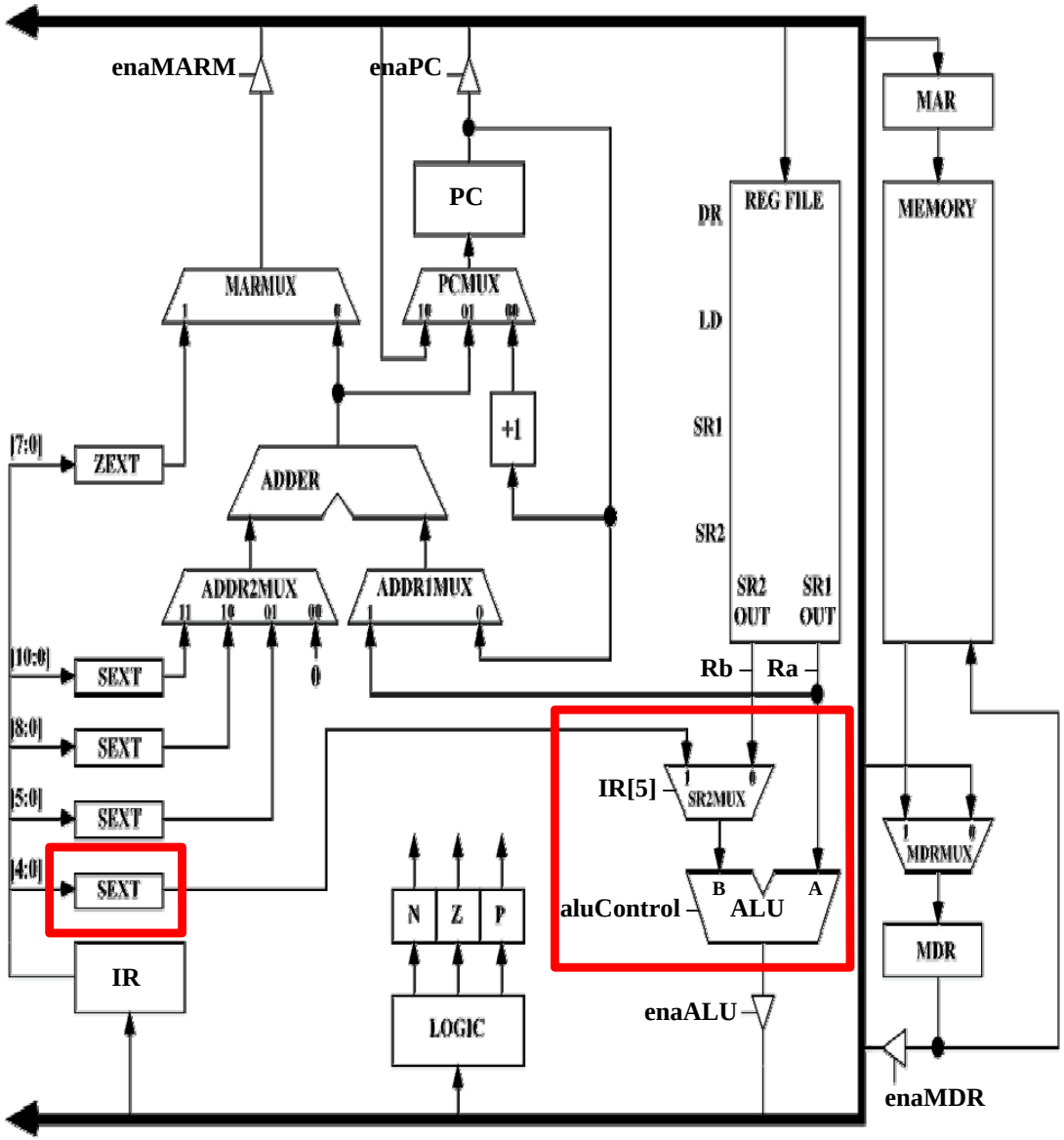


How many of N, Z, P can be high at one time?

The ALU

- Has 4 functions:
  add, and, not, pass
- One operand always comes from regfile, other from mux
- Signals:

  Ra[15:0]
  Rb[15:0]
  IR[4:0]
  IR[5]
  aluControl[1:0]
  aluOut[15:0]



The Operate Instructions

DR = SR1 + SR2

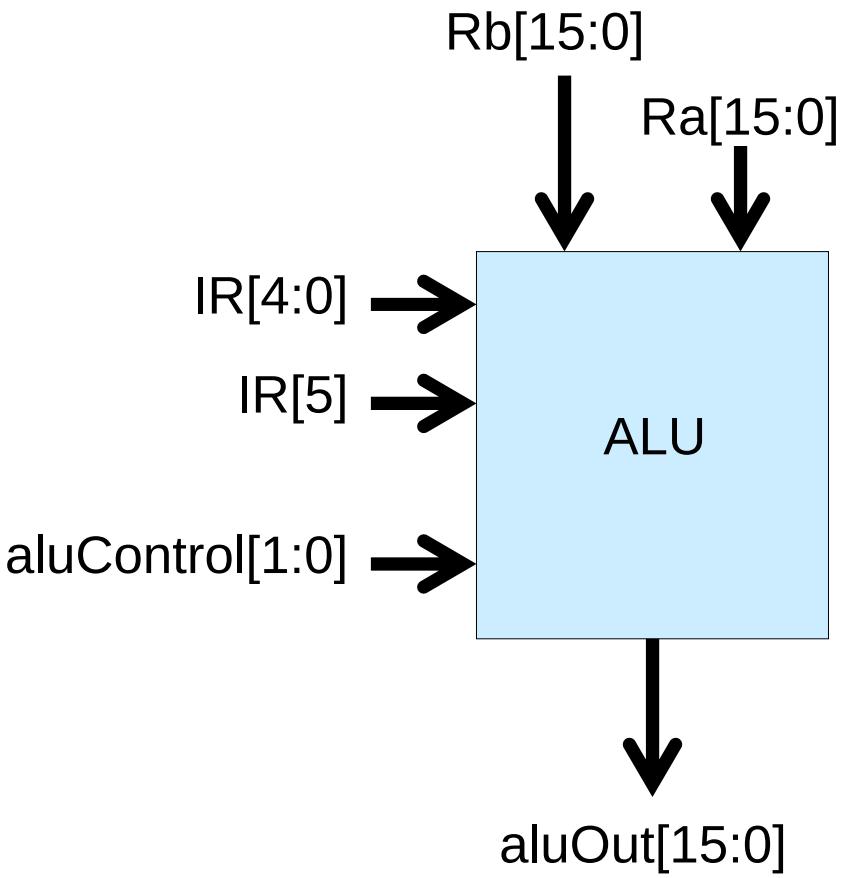
DR = SR1 + SIGNEXTEND(imm5)

DR = SR1 AND SR2

DR = SR1 AND SIGNEXTEND(imm5)

DR = NOT(SR1)

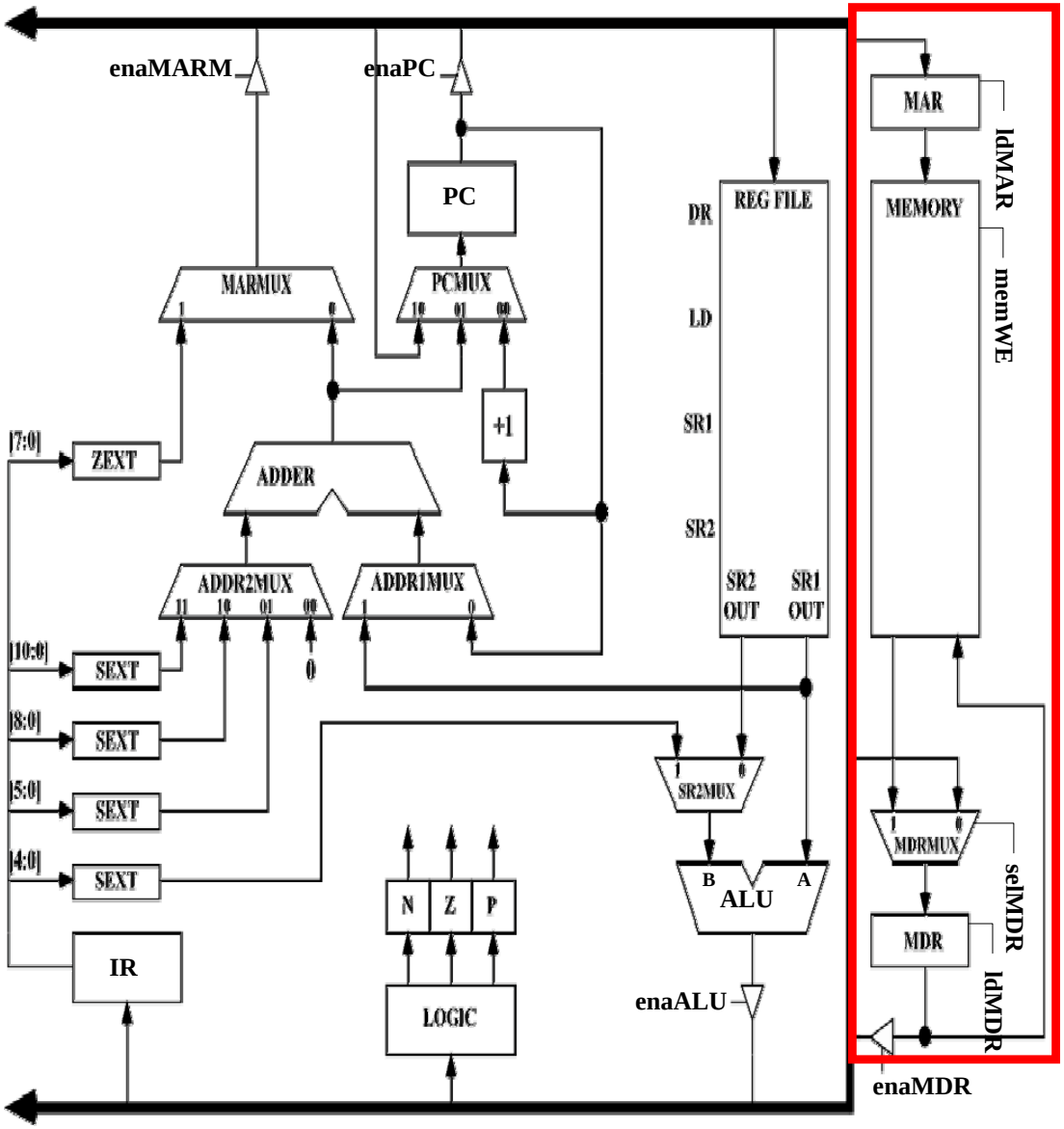
Designing the ALU



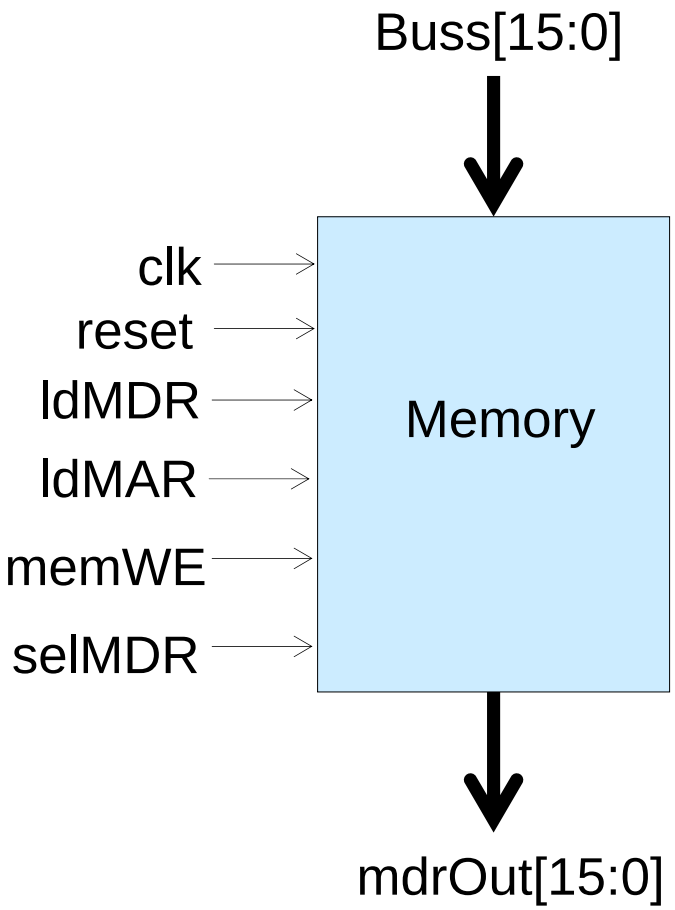
The Memory

- Writes on clock edge when **memWE=1**
- Data read from and written to memory passes through MDR
- The MDRMUX controls where data is loaded from
- Signals:

clk	reset
selMDR	ldMAR
memWE	ldMDR
Buss[15:0]	mdrOut[15:0]



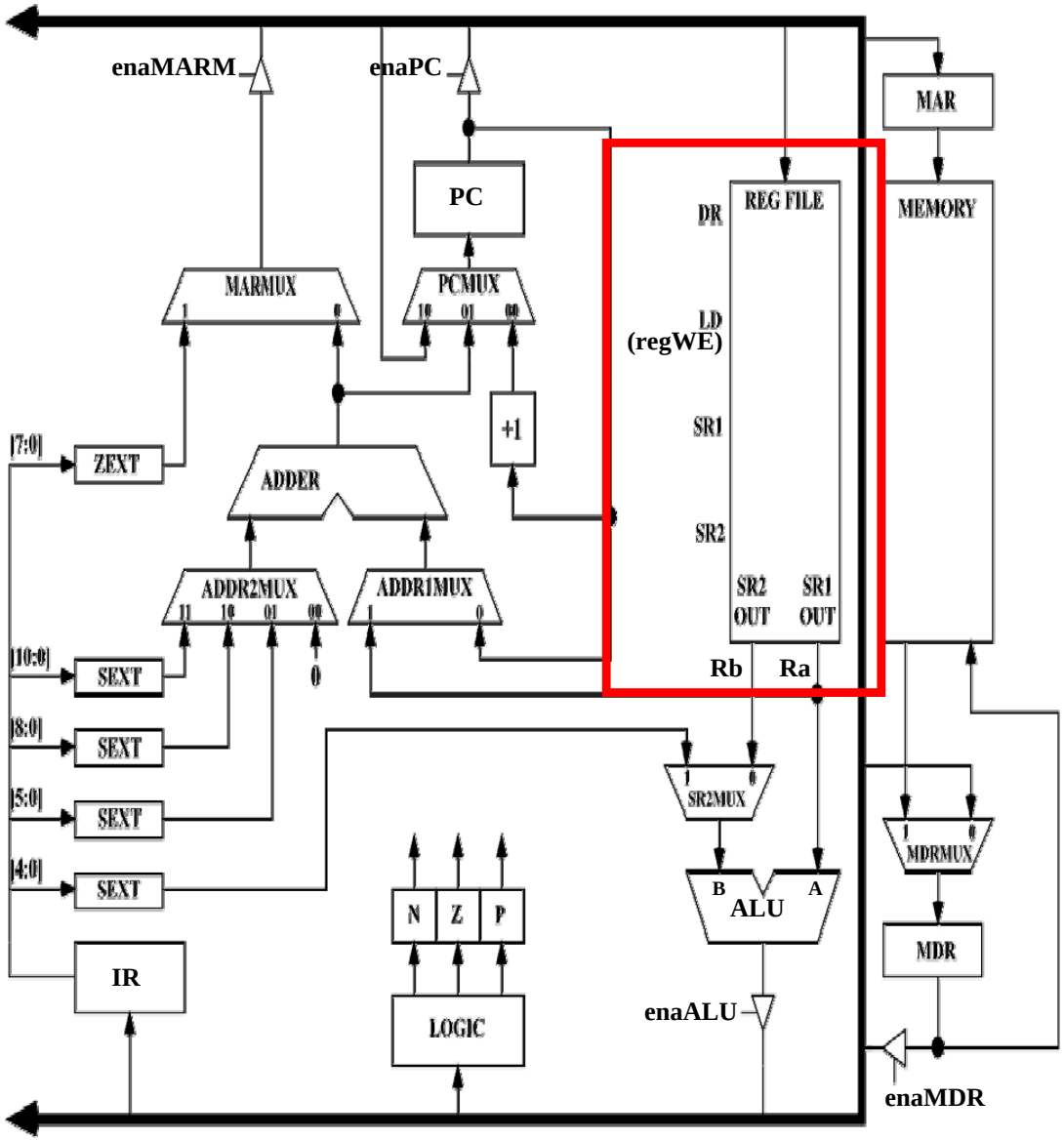
Designing the Memory



The Register File

- Triple-ported memory (2 read ports, 1 write port)
- Synchronous write, asynchronous read
- Signals:

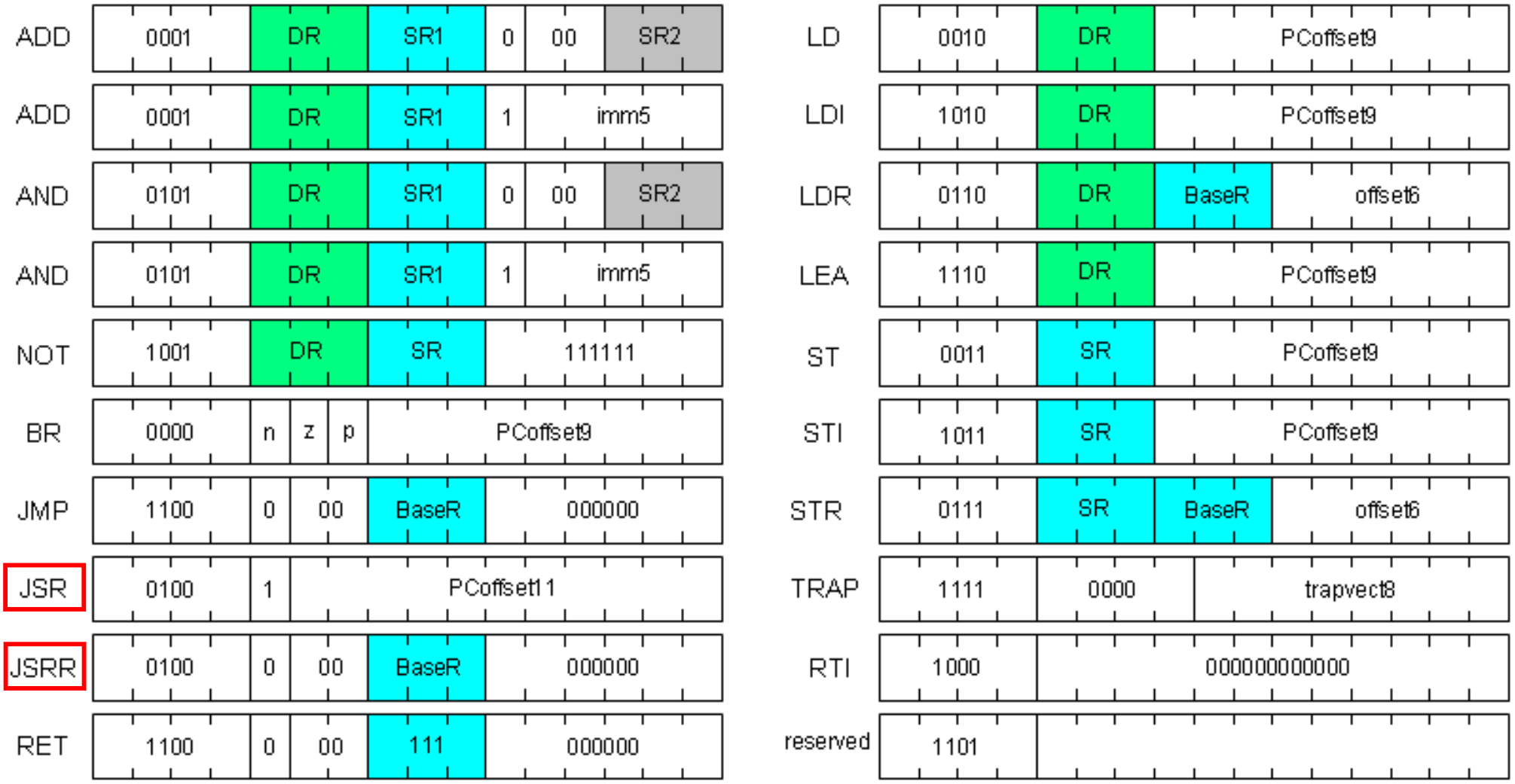
clk	reset
regWE	DR[2:0]
SR1[2:0]	SR2[2:0]
Ra[15:0]	Rb[15:0]
Buss[15:0]	



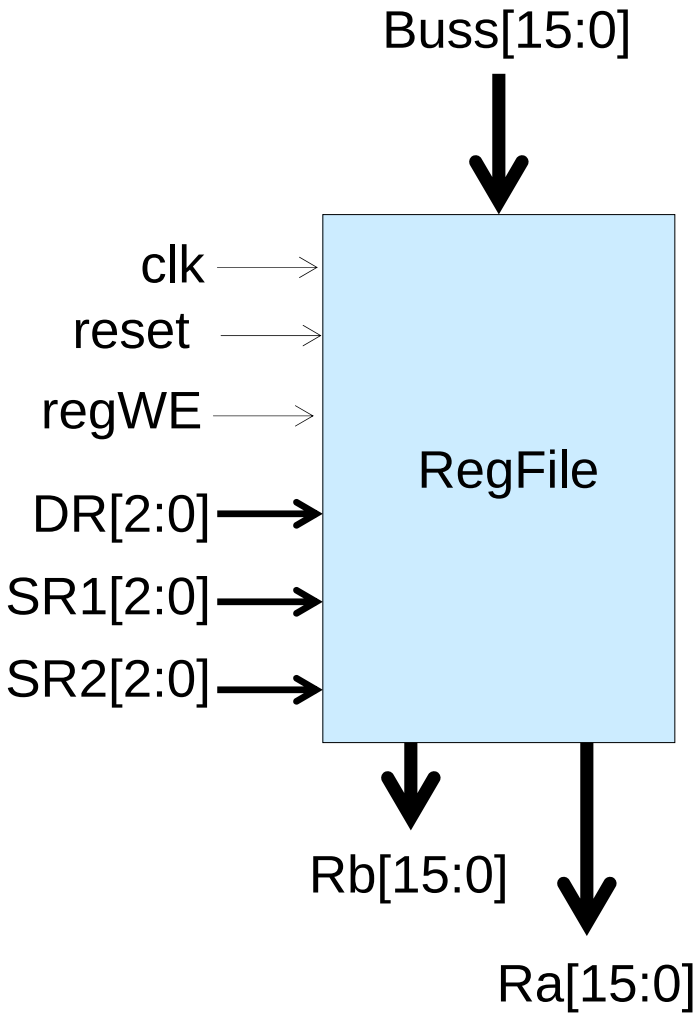
Register File Signals

- **DR** is either "**111**" (subroutine call) or comes from **IR[11:9]**
- **SR2** is always **IR[2:0]**
- **SR1** changes depending on the instruction
  - **IR[11:9]** on **STI, ST**
  - **IR[11:9]** in one state associated with **STR**
  - "**111**" on **RET** (same as **IR[8:6]**)
  - **IR[8:6]** otherwise
- **regWE** is generated by the control block

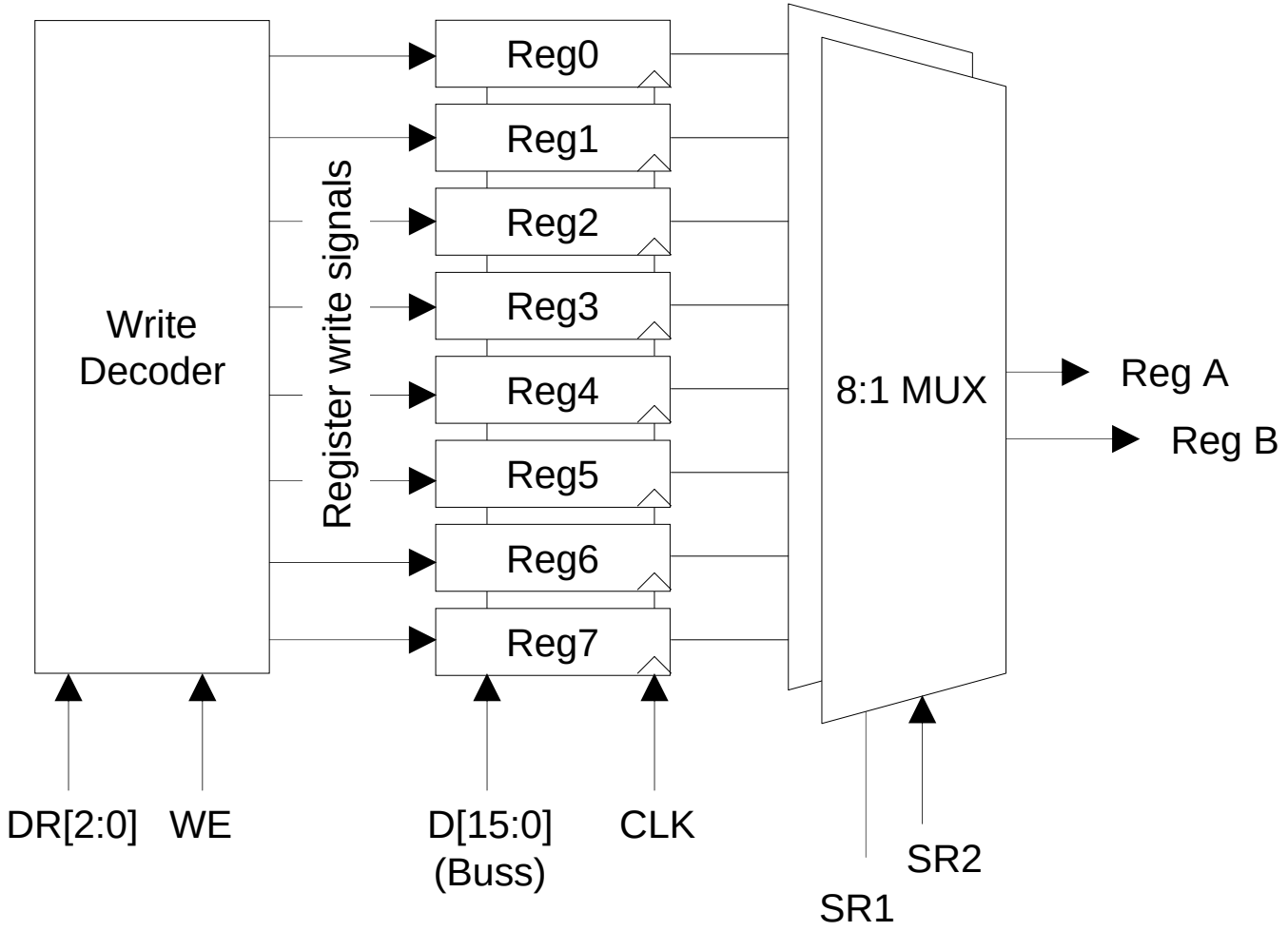
LC-3 Instructions



Building the Register File



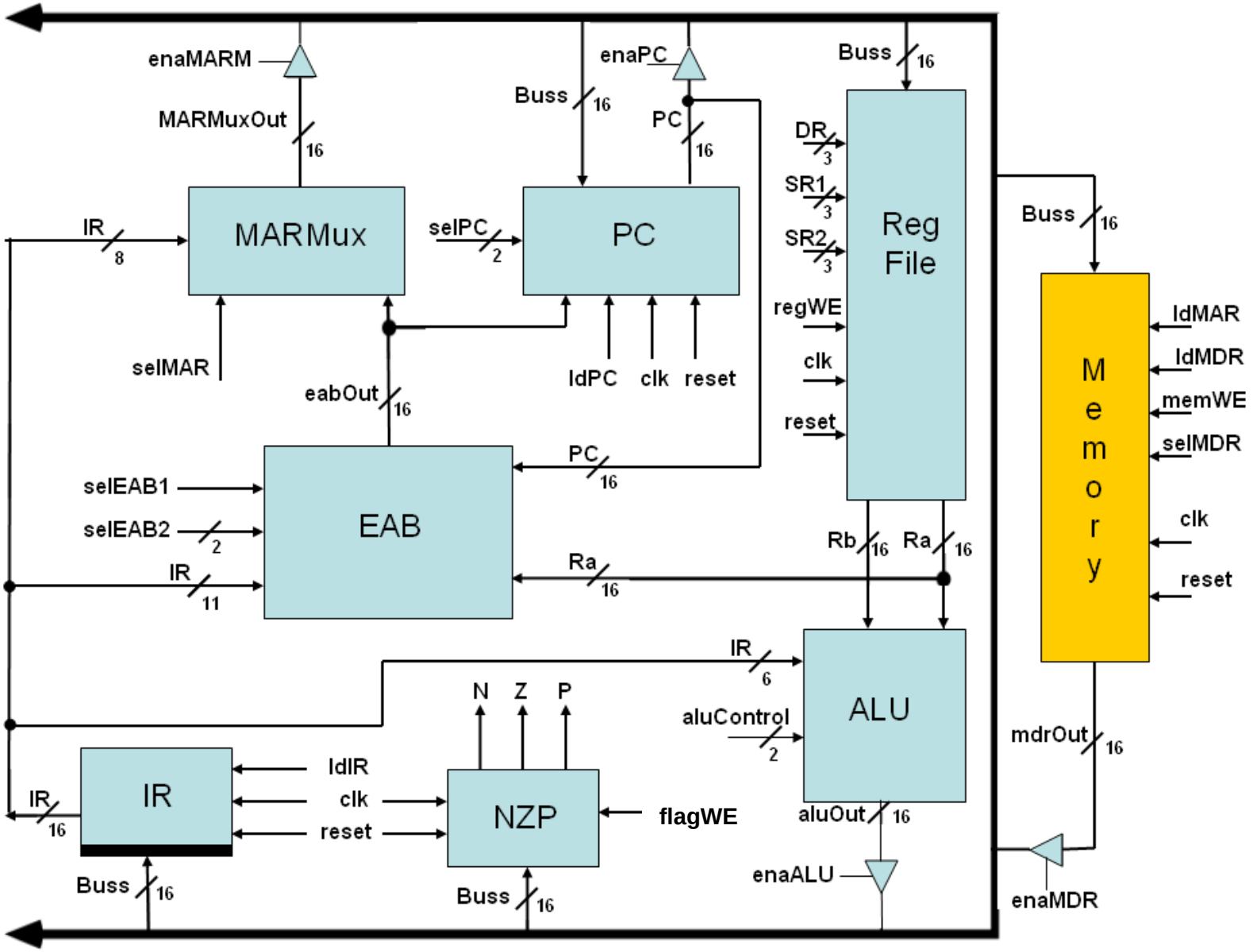
Building the Register File



Building the Machine

- Each element of the datapath can be considered independently, designed independently, and debugged independently
- Once this has been done the control can be easily completed

The LC-3



The LC-3