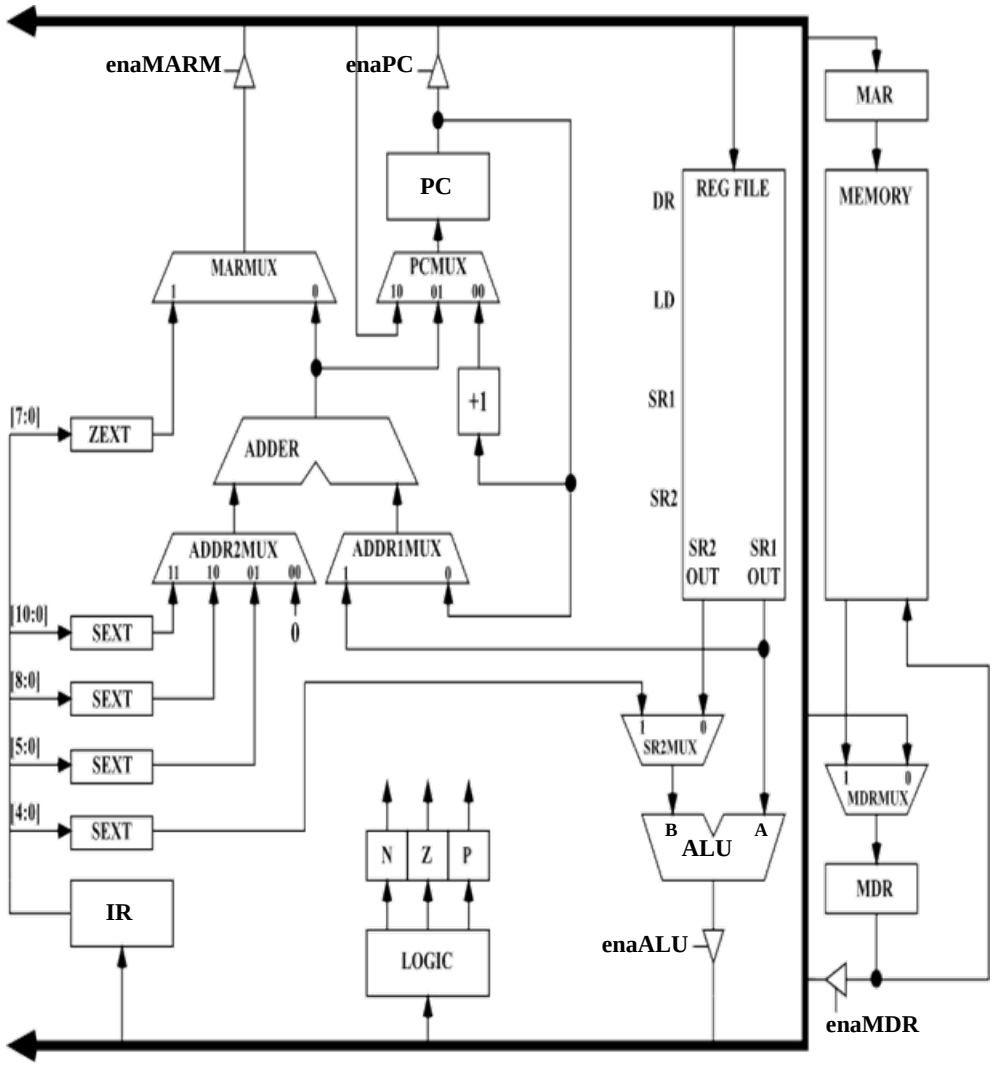
LC3-2
The LC-3 Datapath

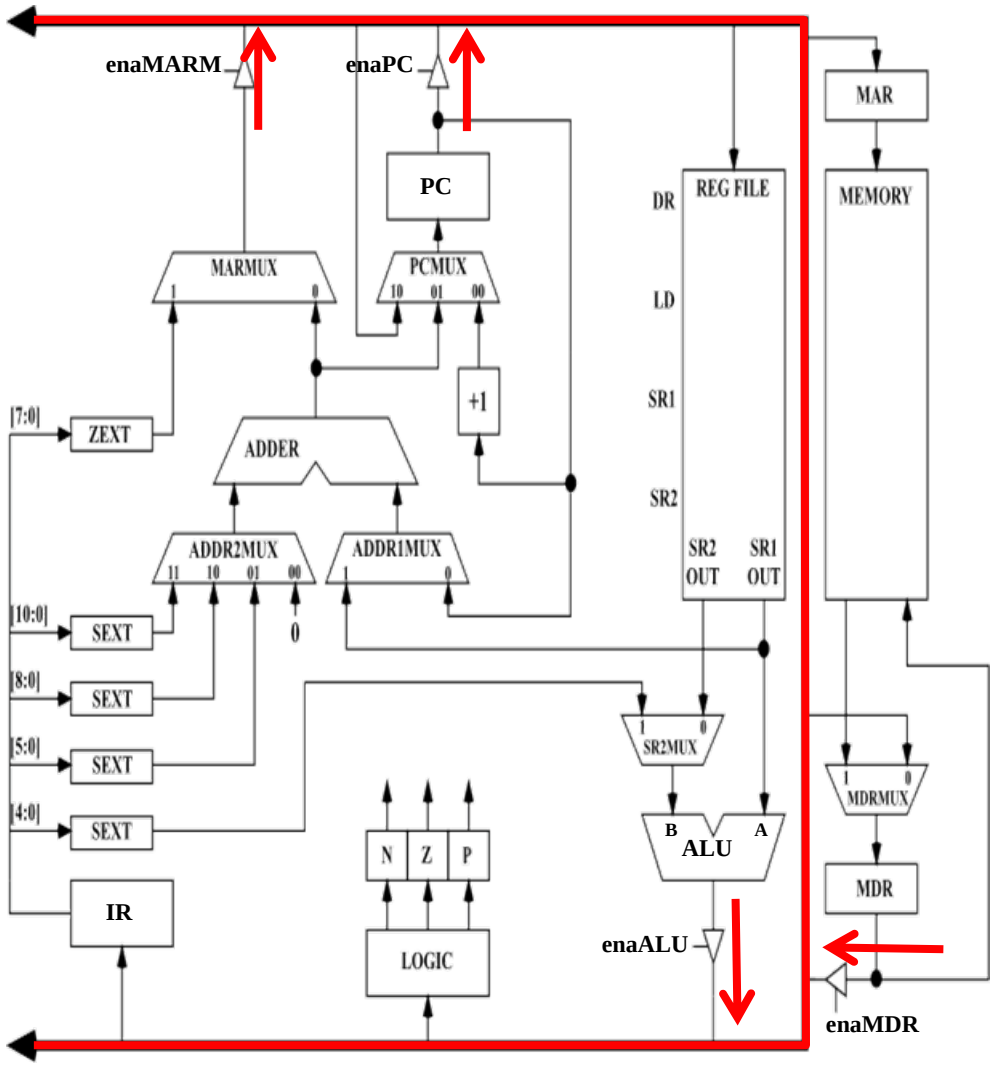


Datapath Strategy

- 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

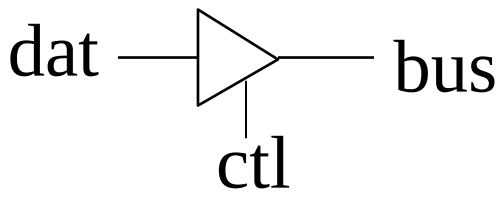
The Bus

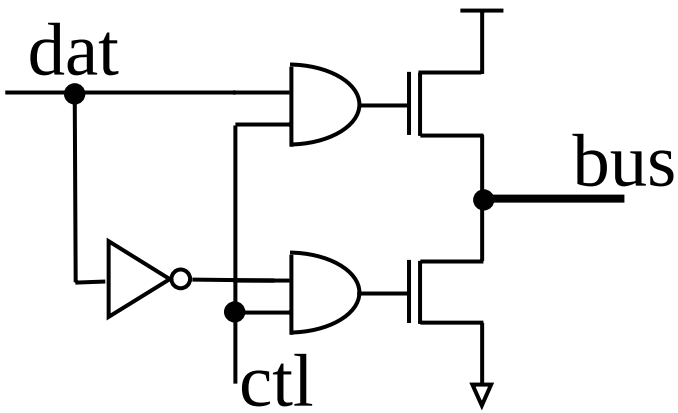
- Four major drivers: MARMux, PC, ALU, MDR
- It is the means of data transfer between units.

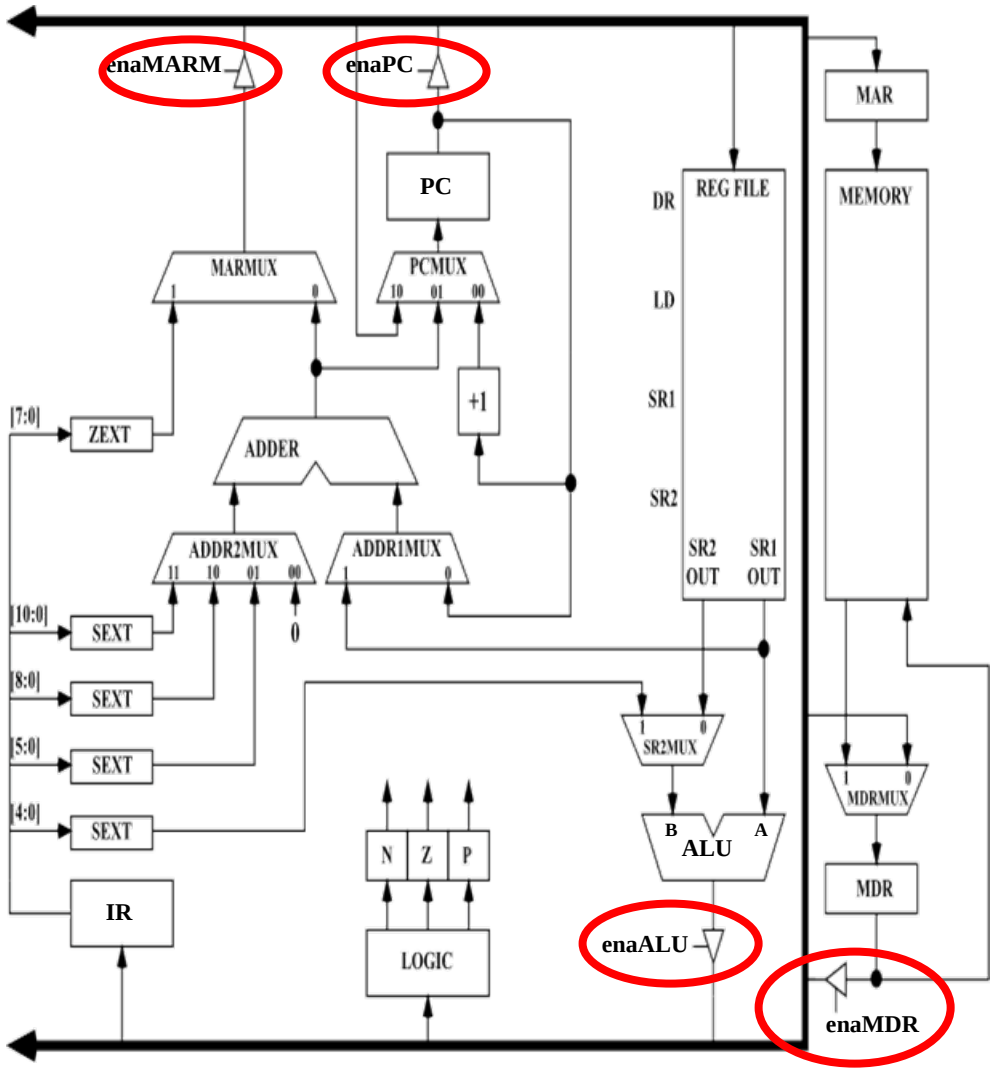


Tri-State Drivers

Bus <= dat when ctl = ‘1’ else ‘ZZZZZZZZZZZZ’;

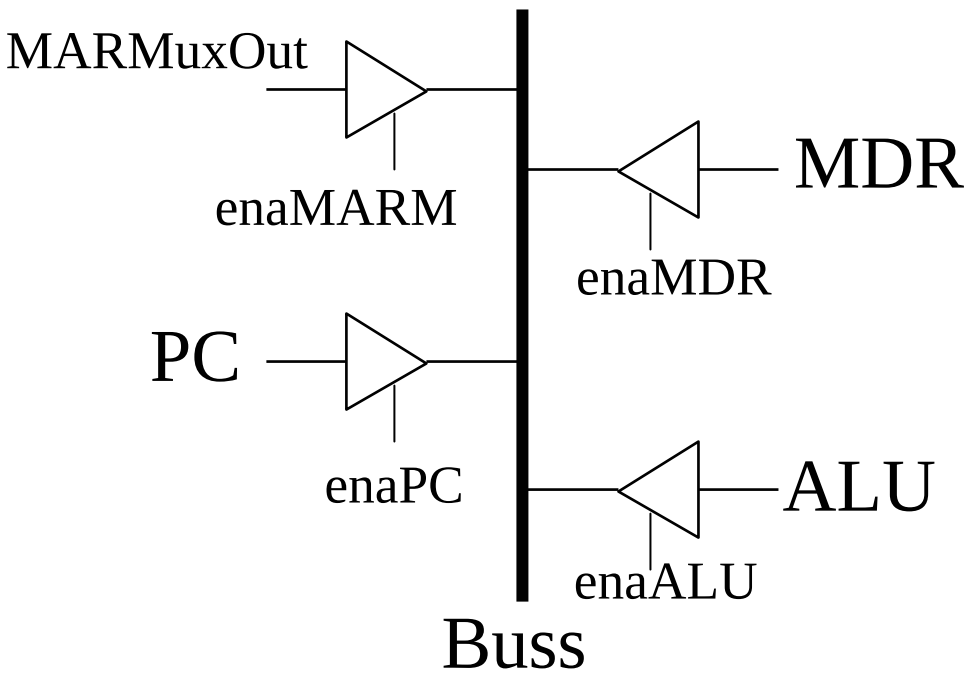






More Tri-State

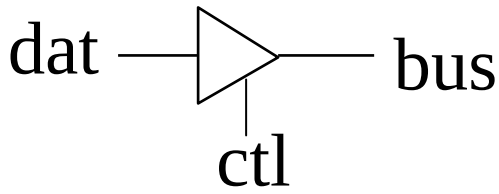
- Lots of drivers and lots of listeners
  - *Only one* driver on at any time
  - Is a *distributed MUX*



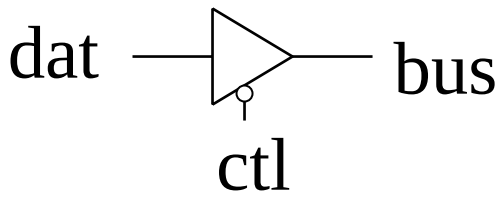
Be careful,
if more than one driver is on at once there will be contention on the bus with unpredictable results

Typical Tri-State Drivers

ctl	dat	bus
0	0	Z
0	1	Z
1	0	0
1	1	1



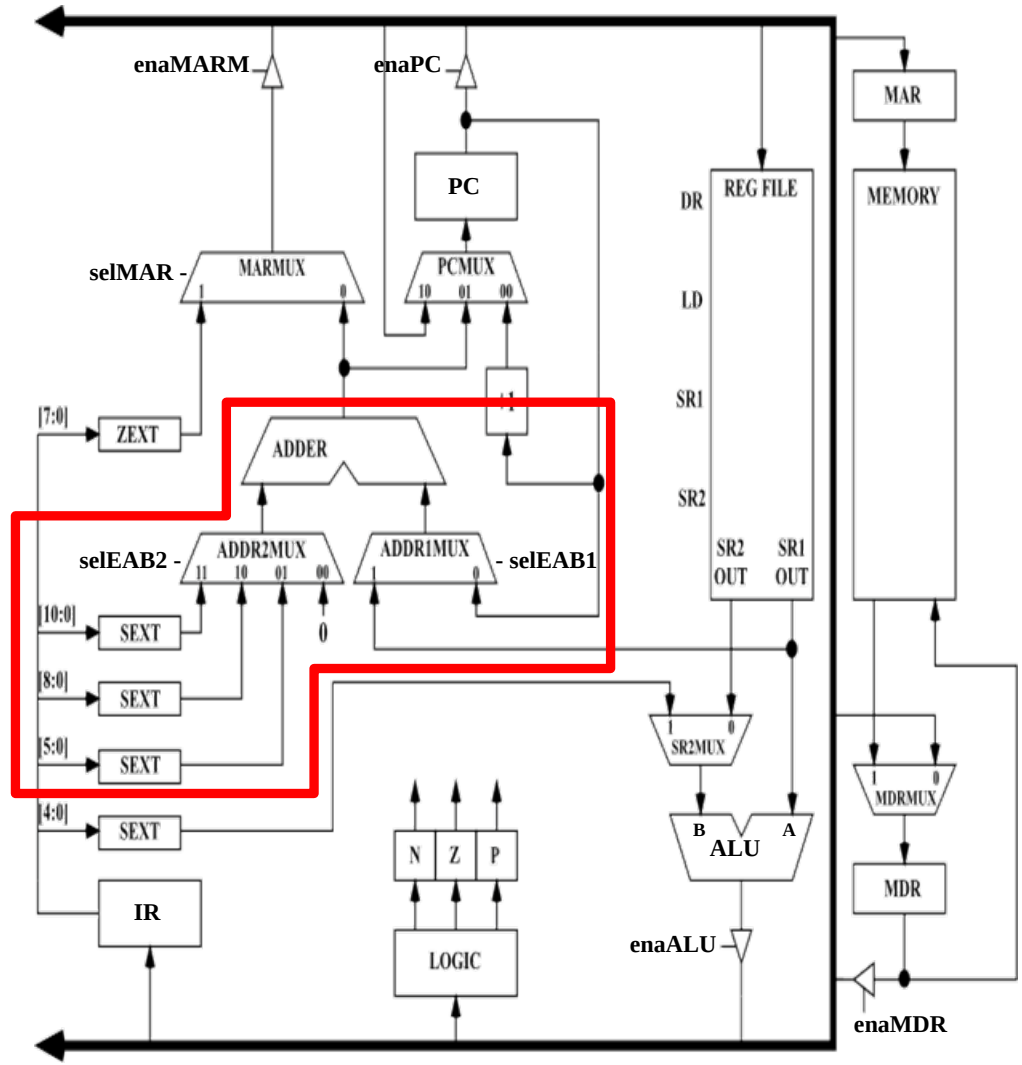
ctl	dat	bus
0	0	0
0	1	1
1	0	Z
1	1	Z



The Effective Address Block

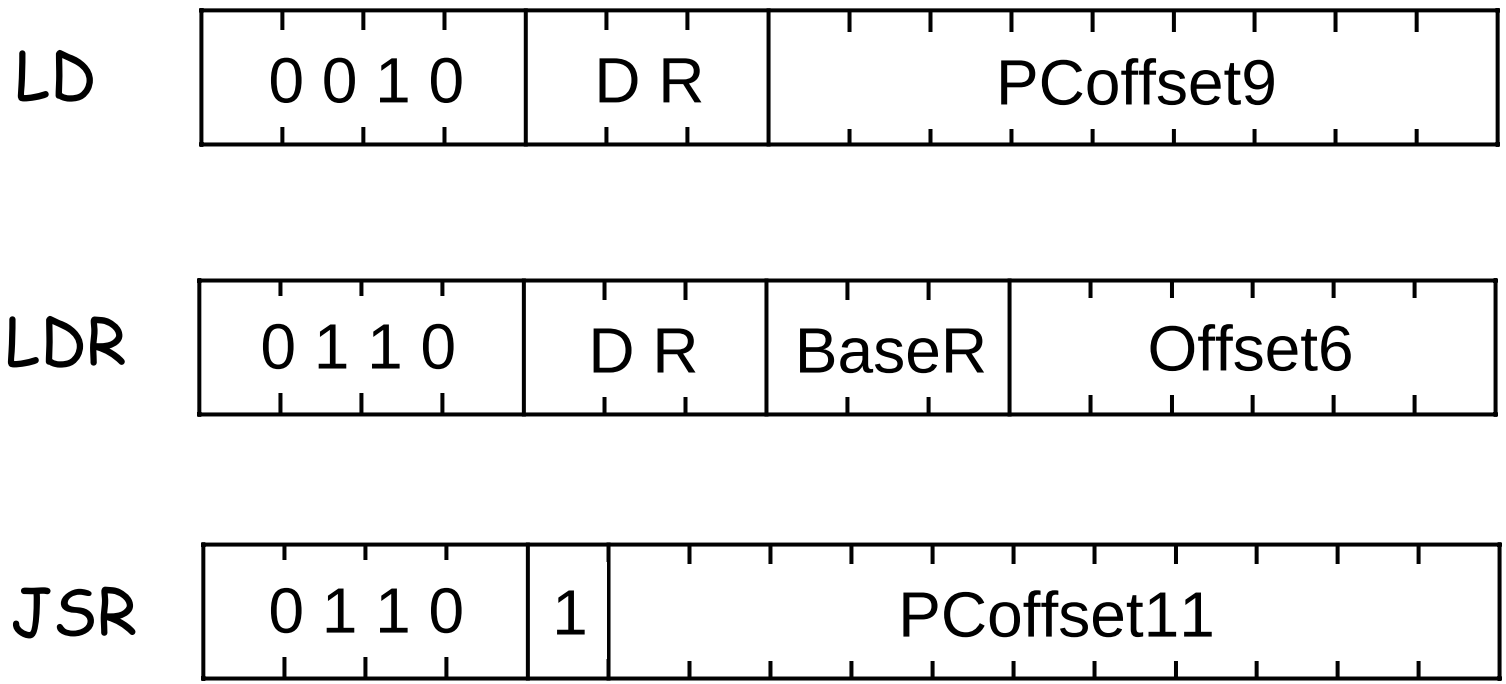
- An adder, two muxes, and three sign extenders
- Signals:

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

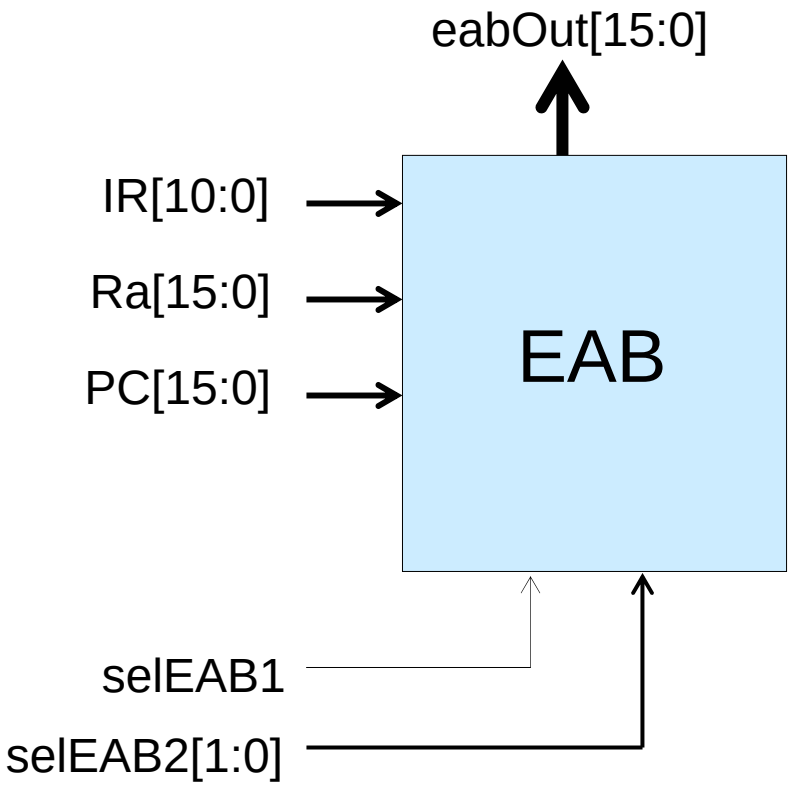


The Effective Address Block

Why are there 3 different sign extenders?



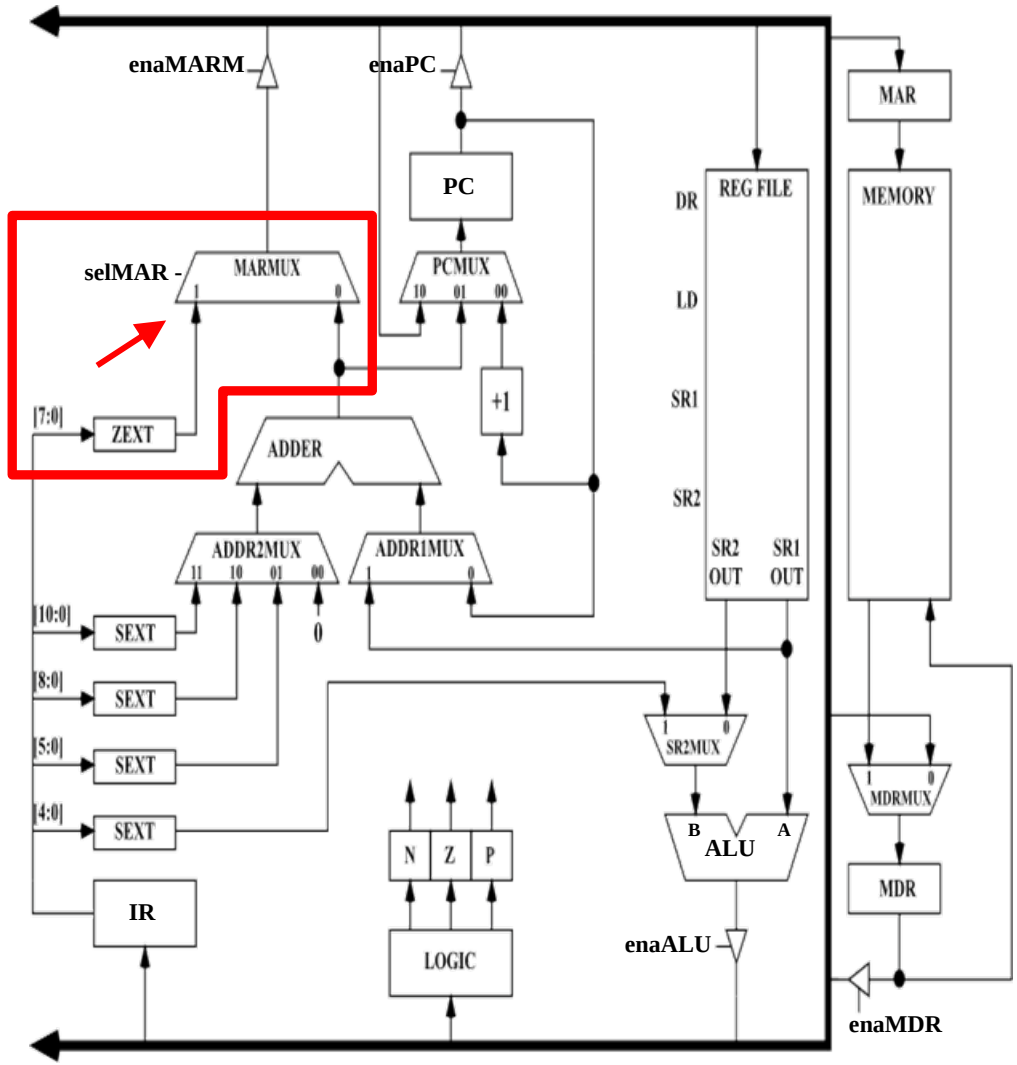
Designing the EAB



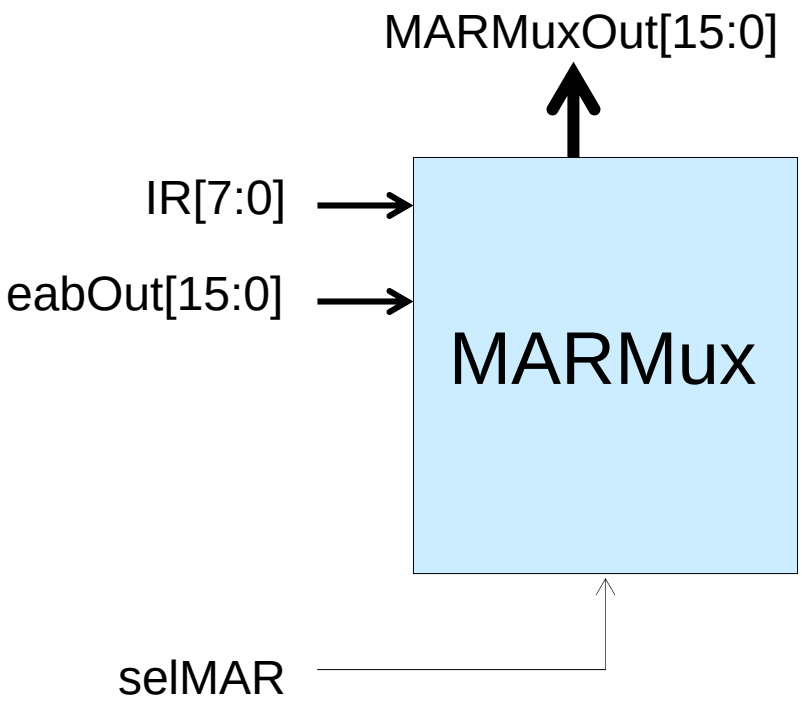
The MARMux

- Simply a MUX and a zero extender
- Signals:

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



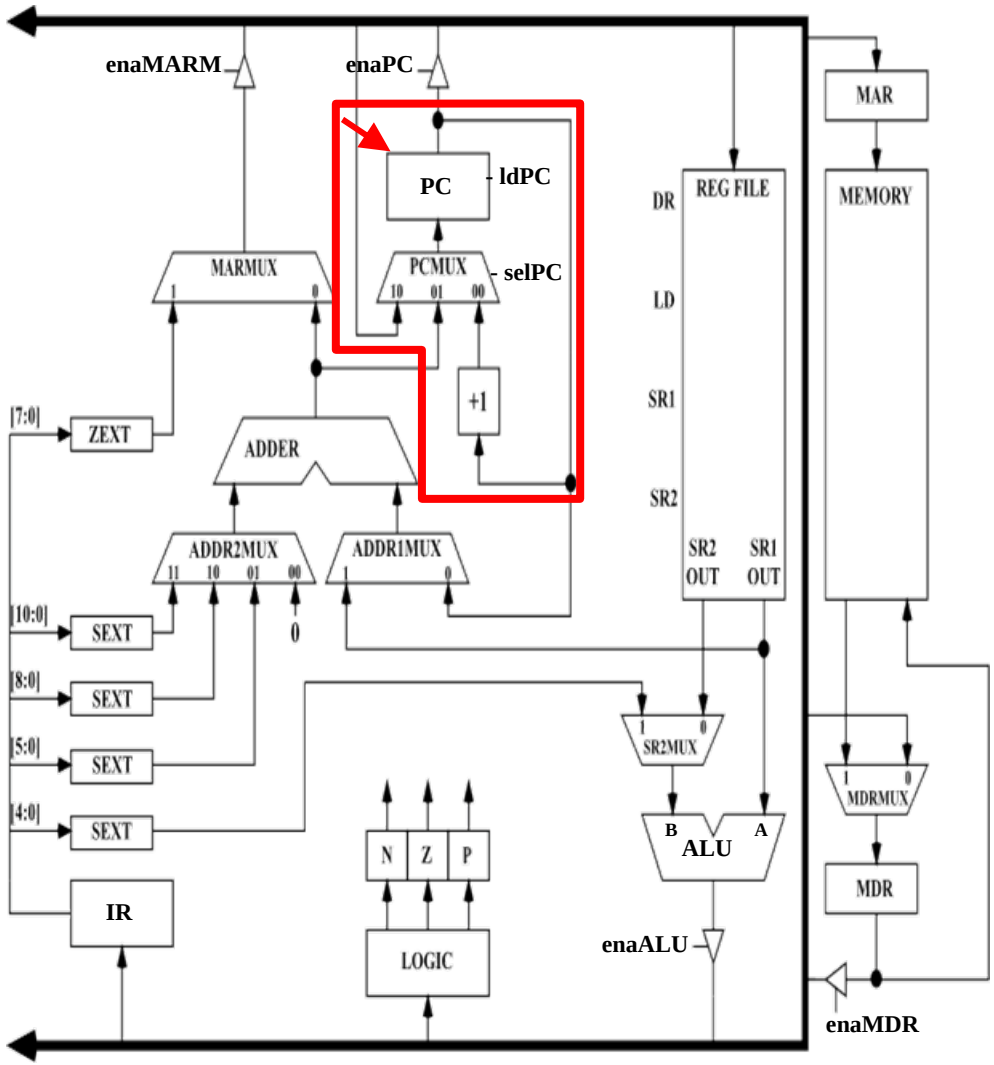
Designing the MARMux



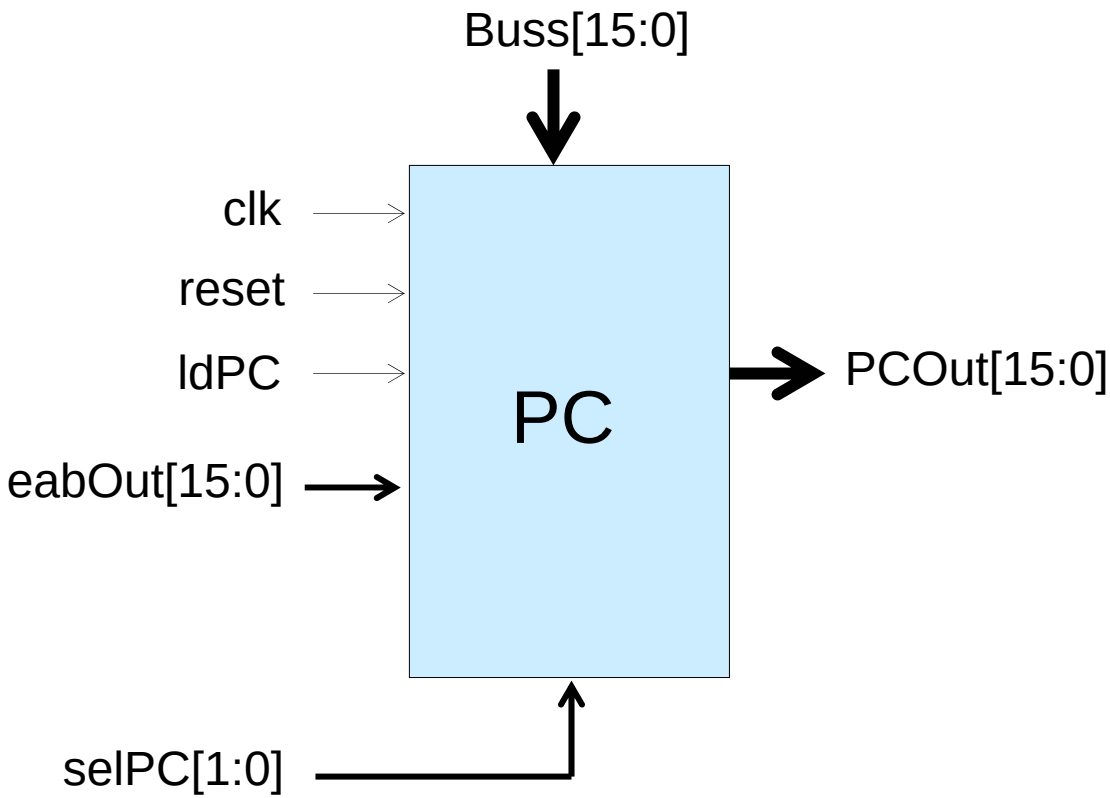
The PC

- 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:

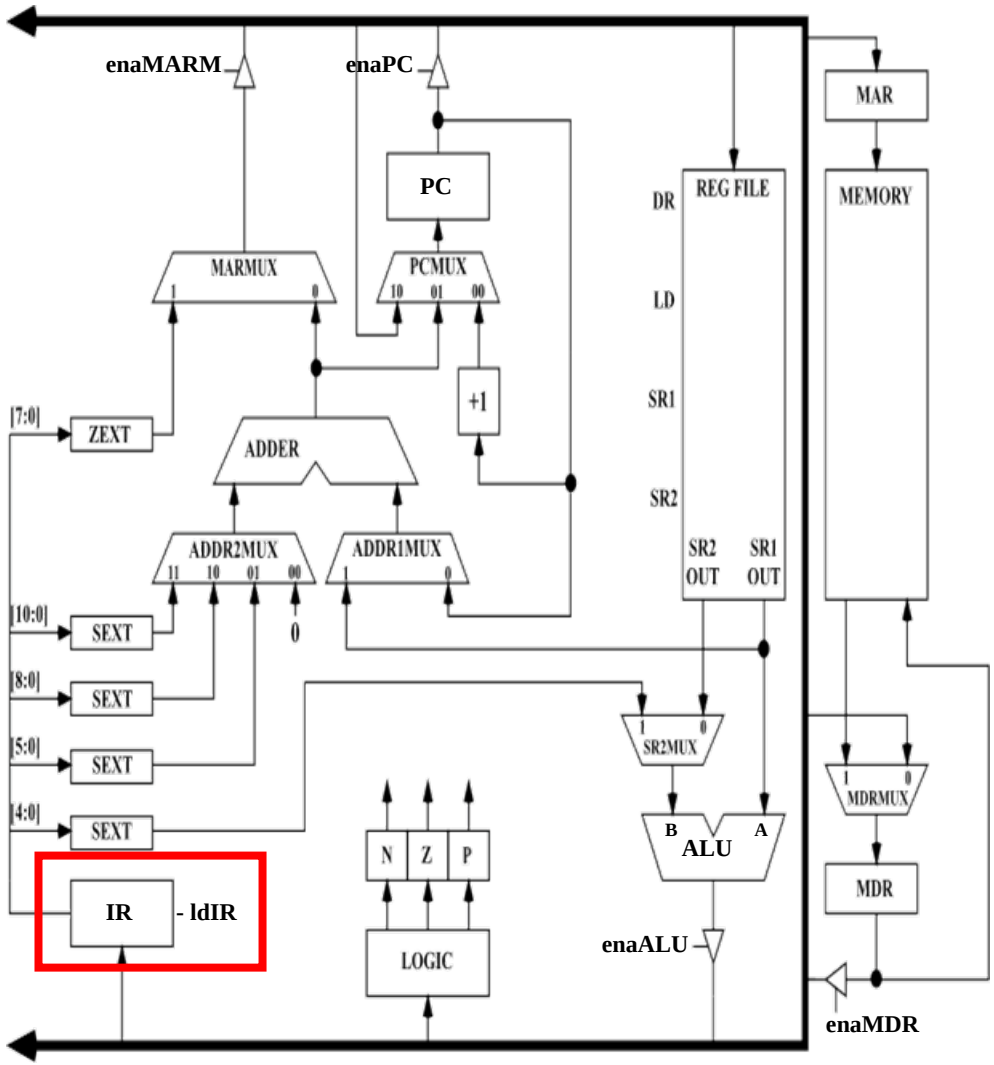
  clk

  reset

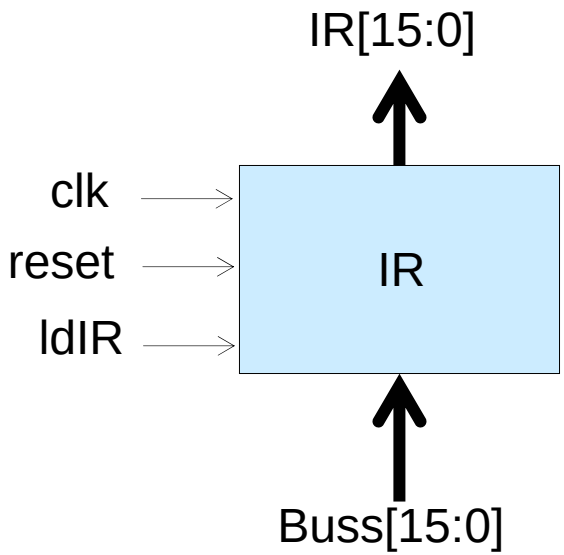
  ldIR

  Buss[15:0]

  IR[15:0]



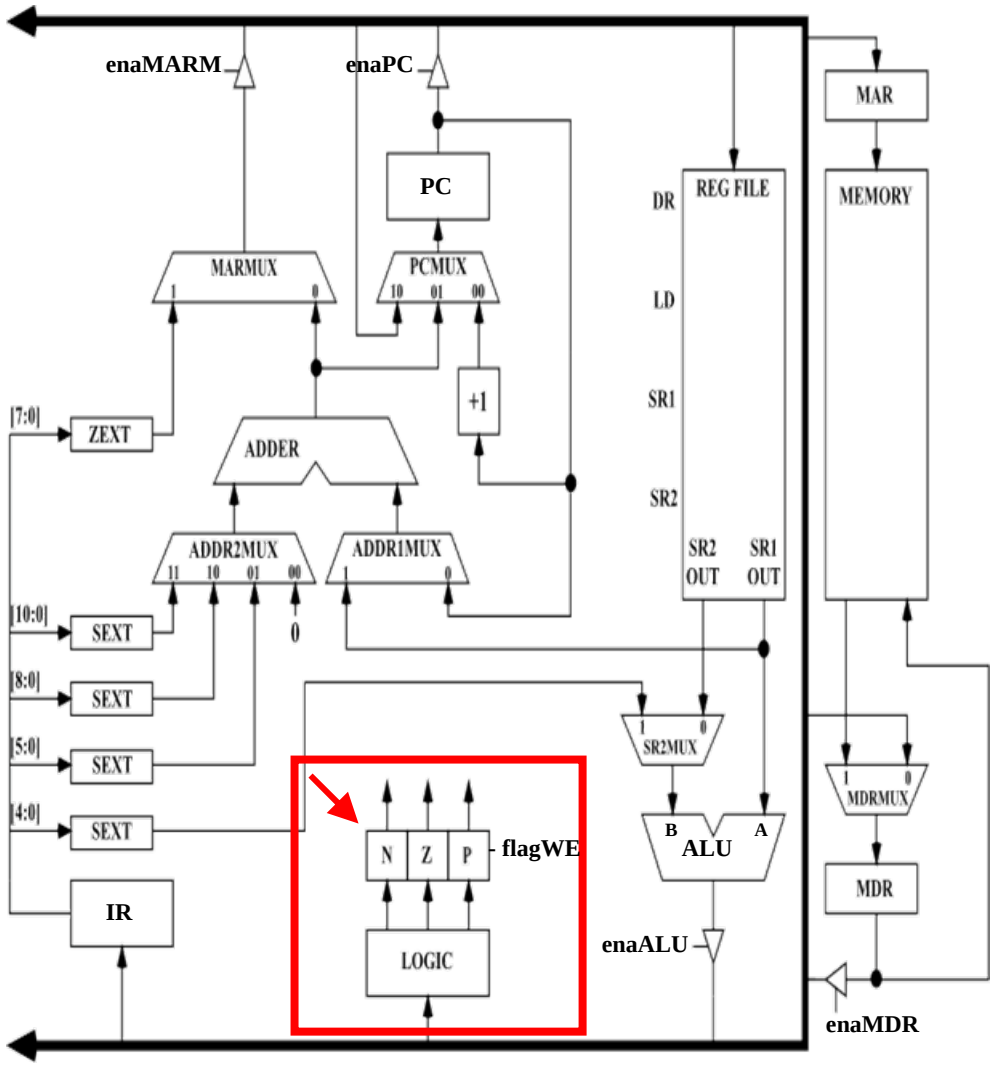
Designing the IR



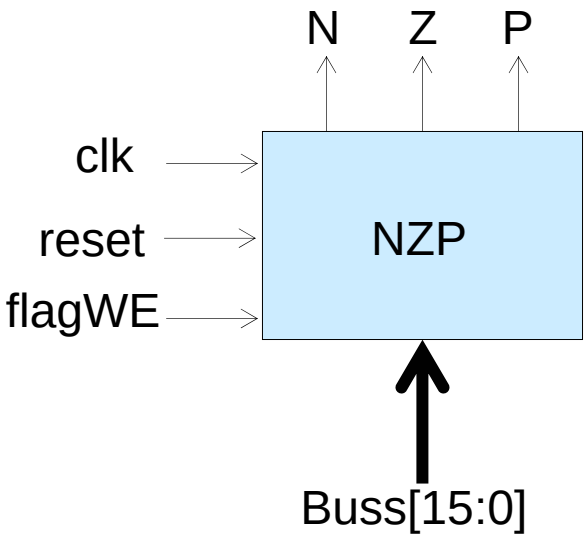
The nzp Flags

- 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

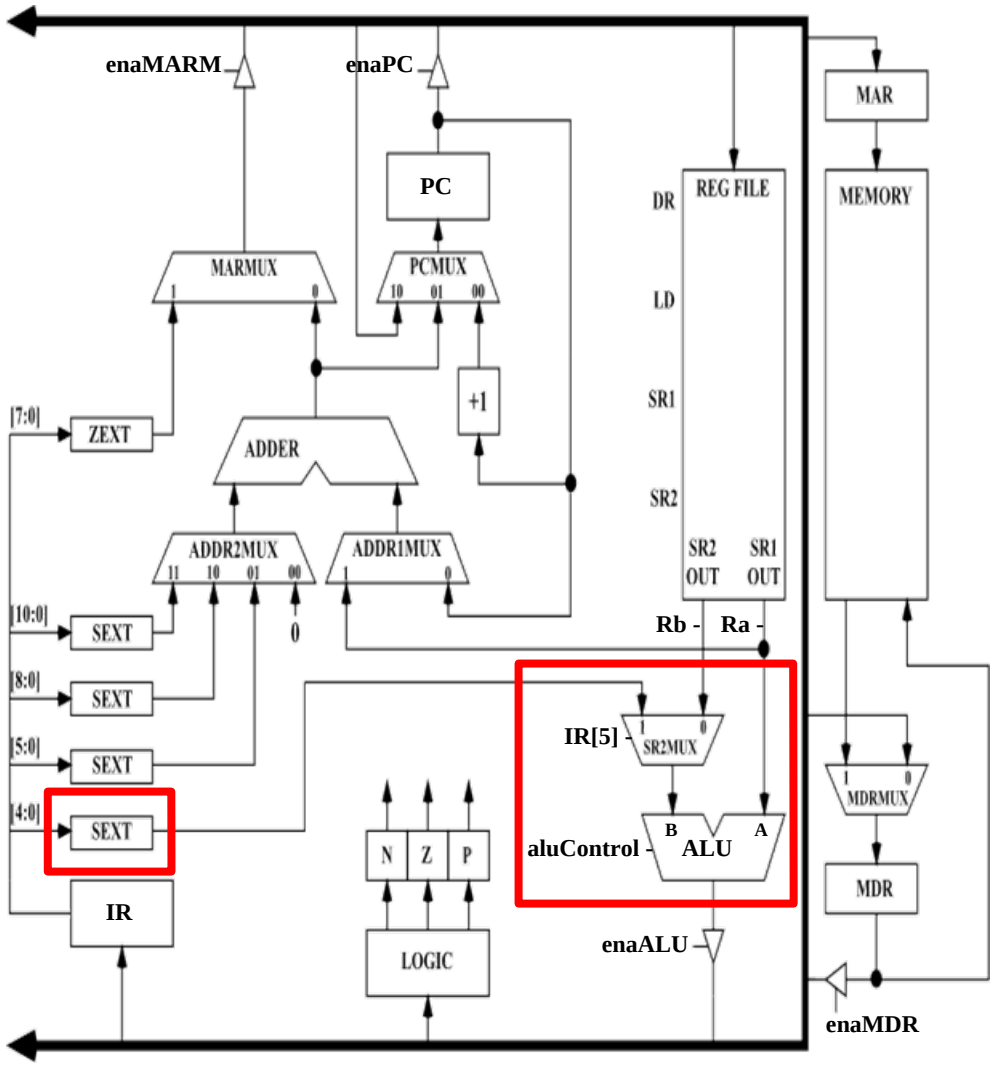


Is it obvious to you that one of N,Z,P is always high while the other two are low?

The ALU

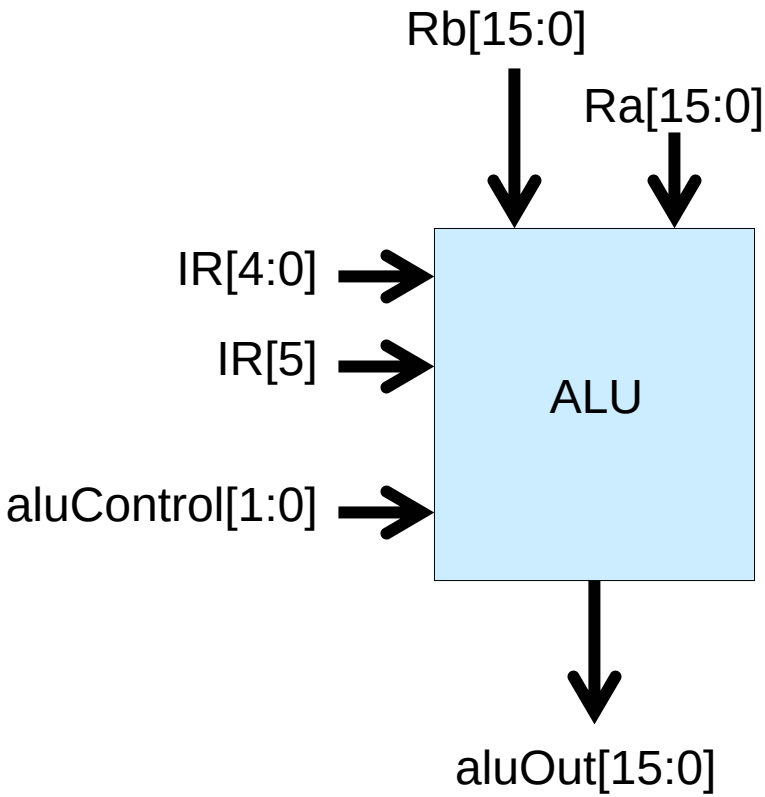
- Has 4 functions:
  add, and, not, pass
- One operand always comes from regfile, other from mux
- Signals:

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



The Operate Instructions

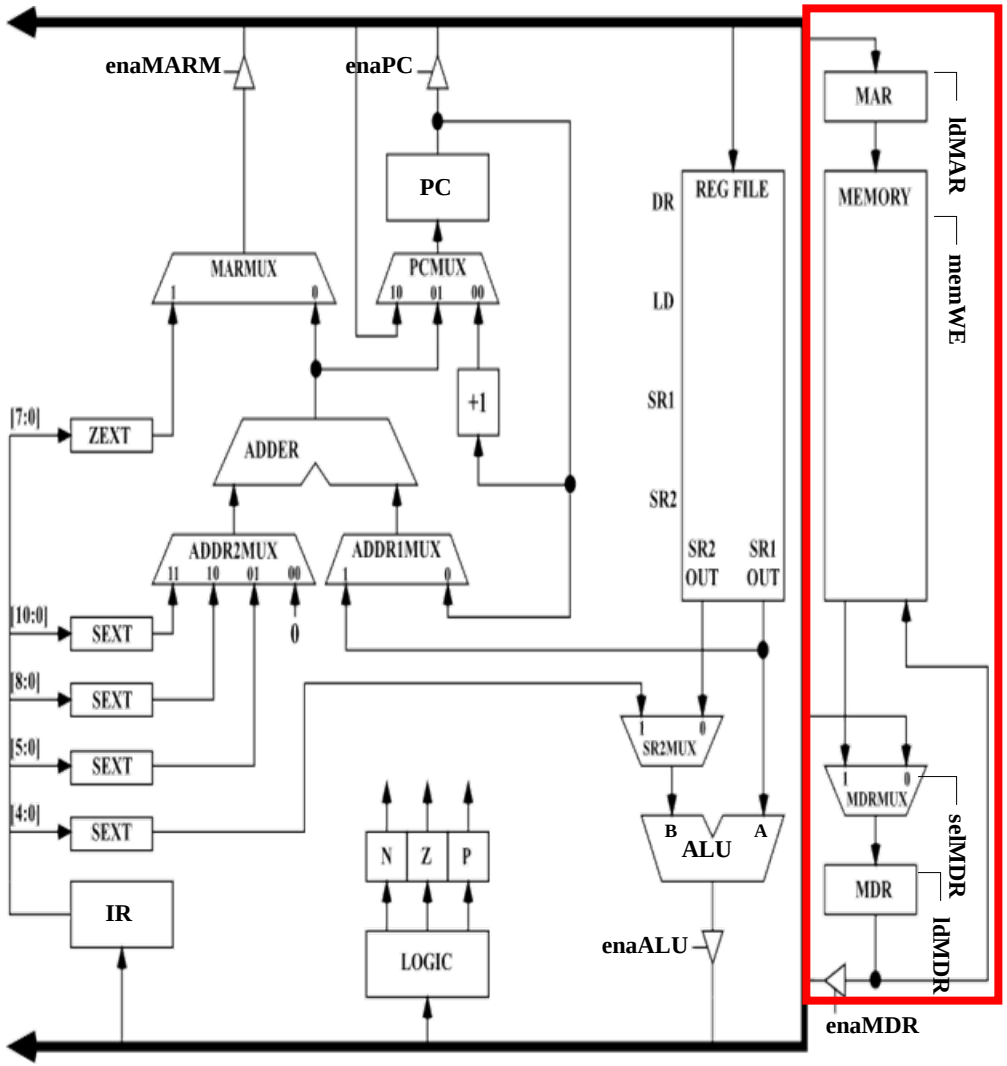
Designing the ALU



The Memory

- Writes on clock edge when **memWE=1**
- Data read from and written to memory is first loaded into the MDR
- The MDRmux controls where data is loaded from
- Signals:

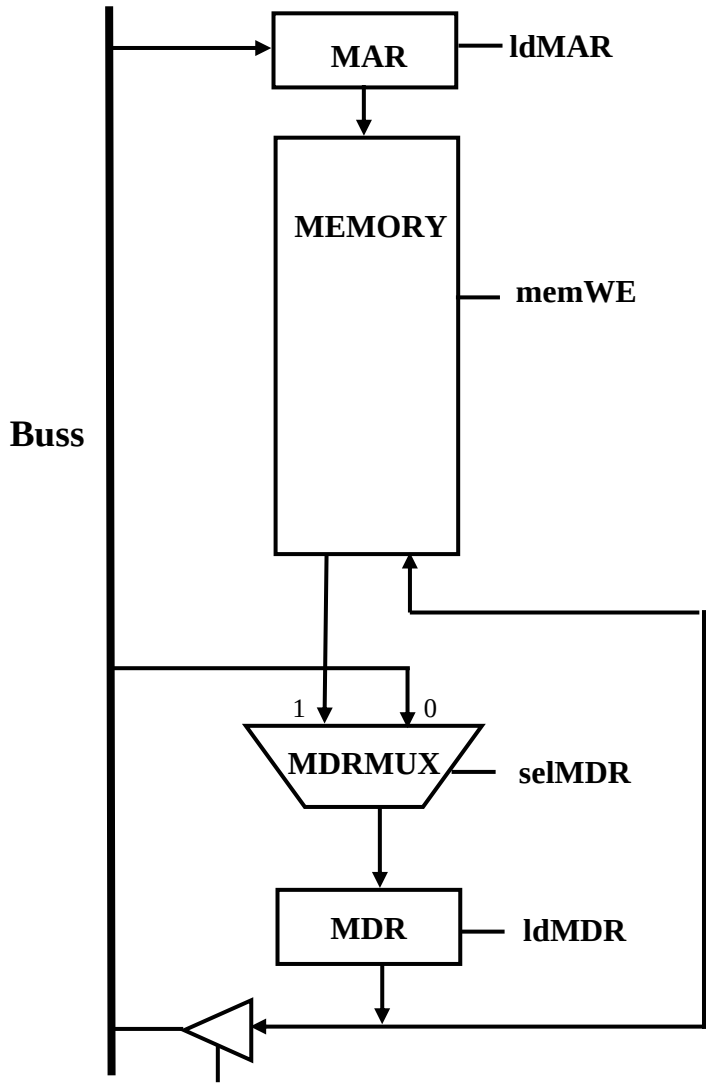
clk selMDR
ldMAR Buss[15:0]
ldMDR mdrOut[15:0]
memWE



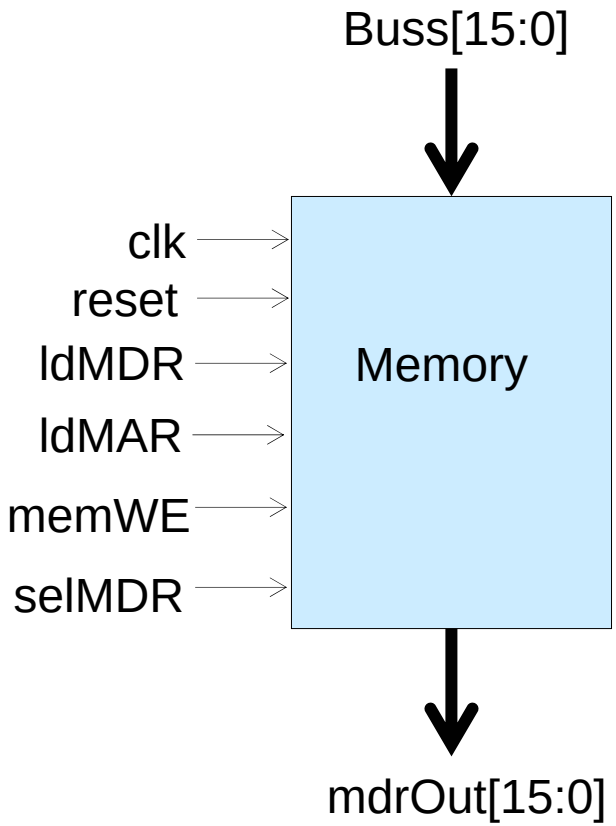
The Memory

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

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



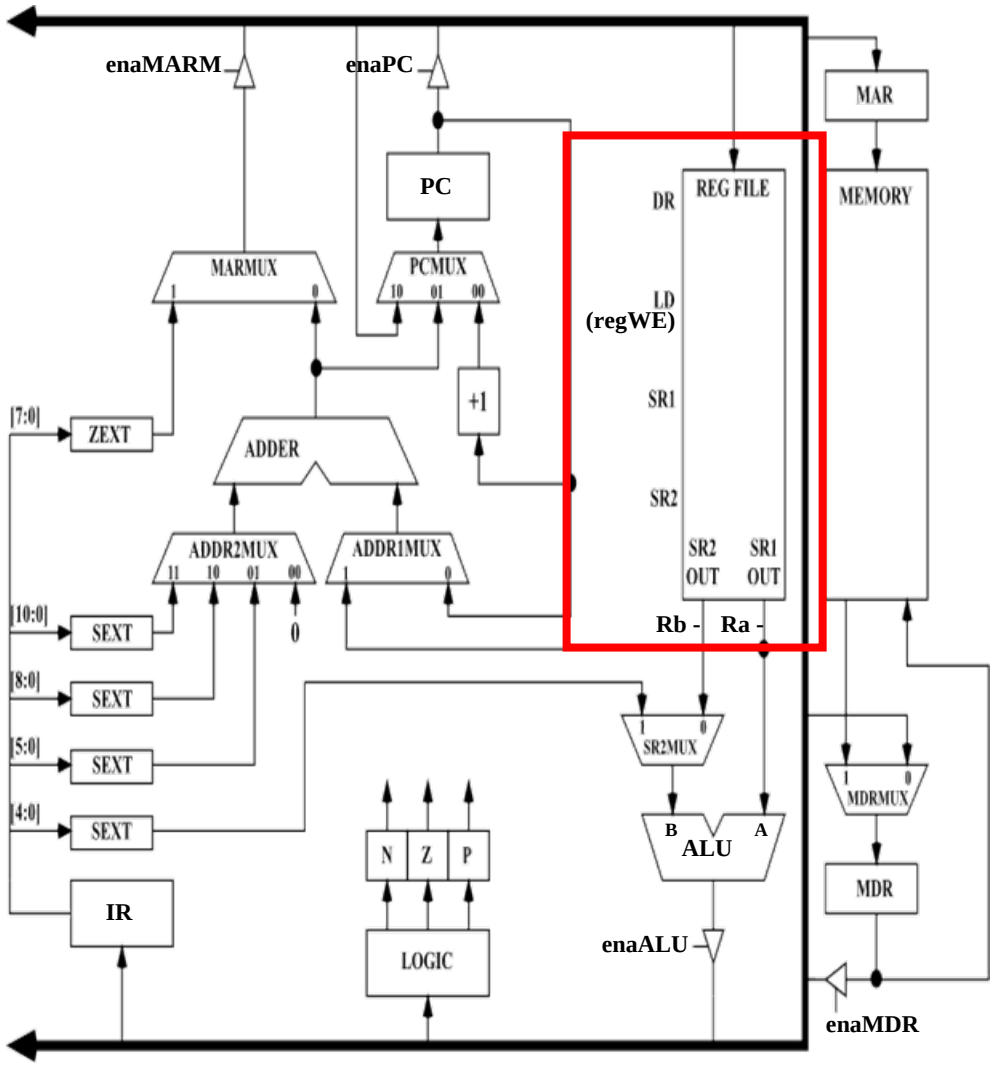
Designing the Memory



The Register File

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

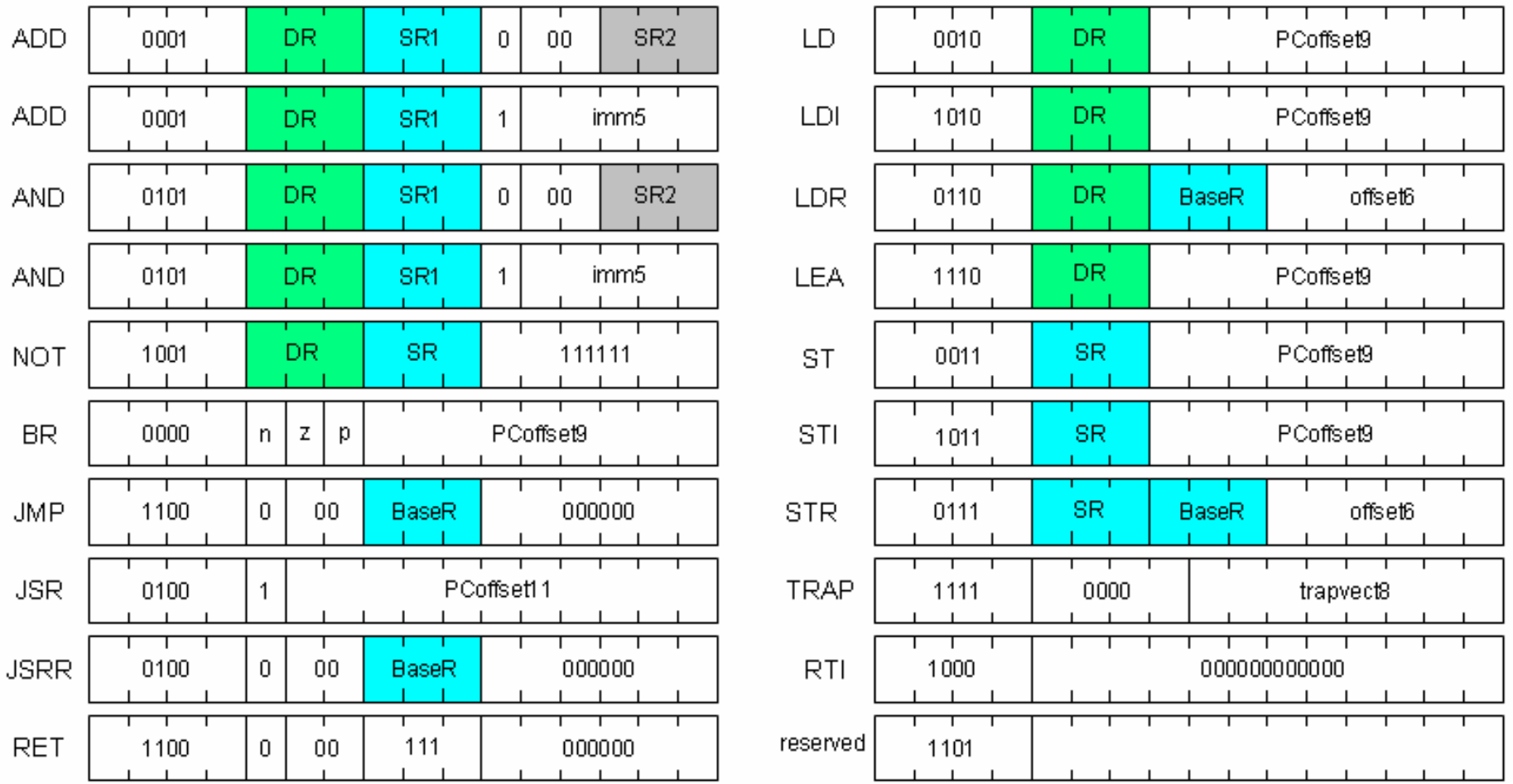
  clk
  reset
  regWE
  DR[2:0]
  SR1[2:0]
  SR2[2:0]
  Ra[15:0]
  Rb[15:0]
  Buss[15:0]



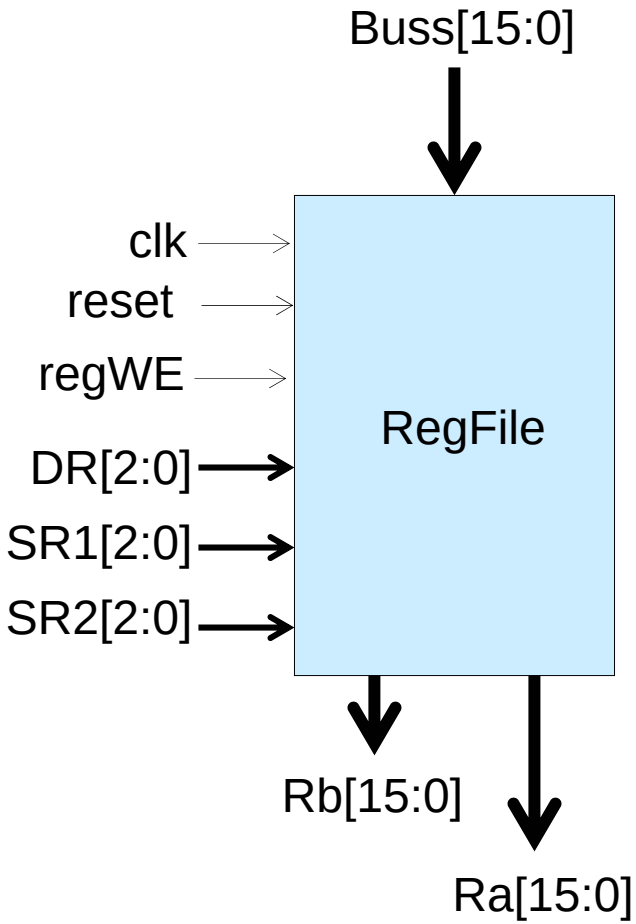
Register File Signals

- **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
  - **IR(8-6)** otherwise
- **regWE** is generated by the control block

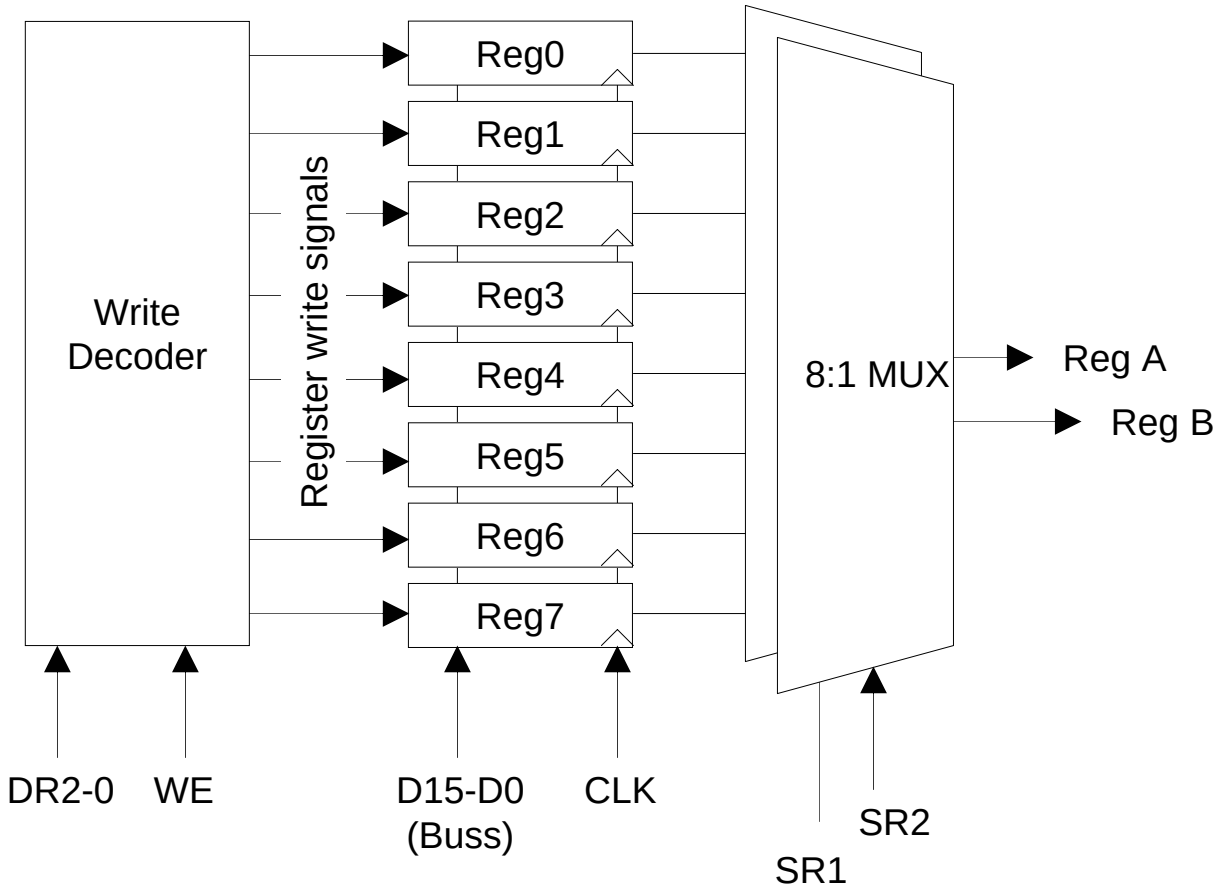
LC-3 Instructions



Building the Register File



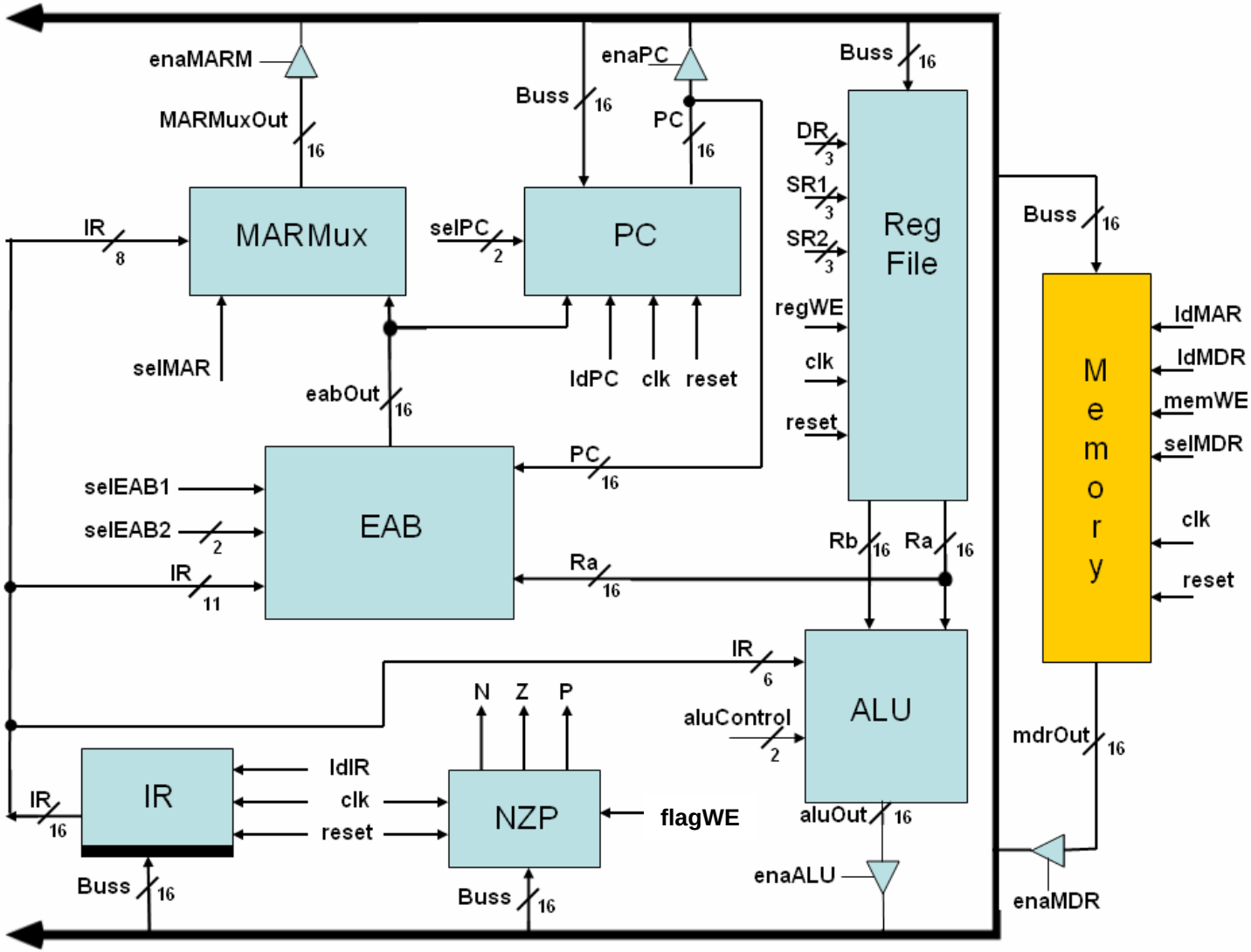
Building the Register File



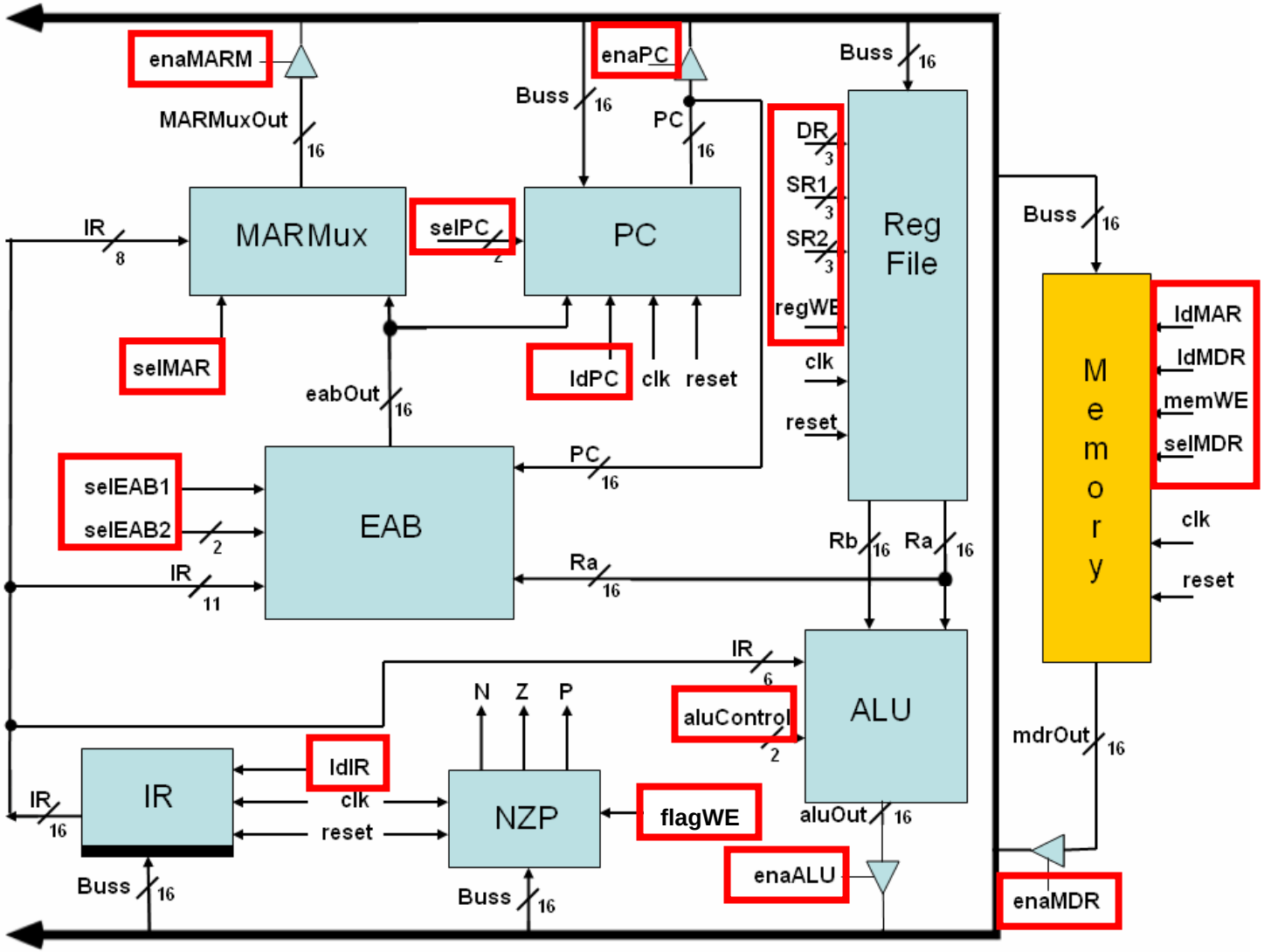
Building the Machine

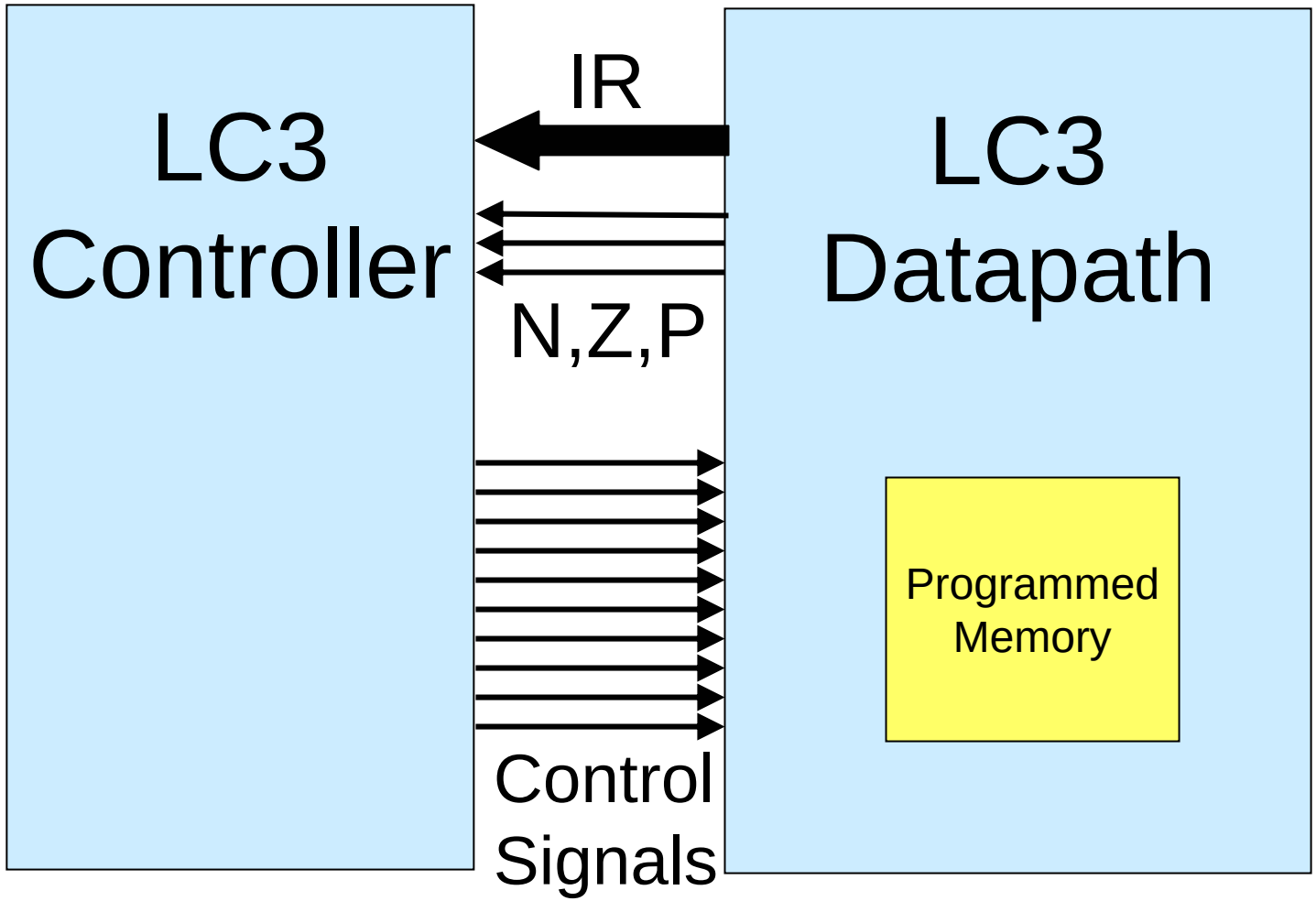
- Each element of the datapath can be considered independently, designed independently, and debugged independently
- Once this has been done the control can be easily completed

The LC-3


enaMARM
MARMuxOut
16
Buss
16
enaPC
PC
16
Buss
16
DR
3
SR1
3
SR2
3
Reg File
regWE
clk
reset
Buss
16
IR
8
MARMux
selPC
2
PC
selMAR
eabOut
16
ldPC
clk
reset
PC
16
selEAB1
selEAB2
2
IR
11
EAB
Ra
16
Rb
16
Ra
16
IR
6
N
Z
P
aluControl
2
ALU
aluOut
16
ldIR
clk
reset
IR
16
IR
NZP
flagWE
Buss
16
Buss
16
enaALU
ldMAR
ldMDR
memWE
selMDR
clk
reset
Memory
mdrOut
16
enaMDR

The LC-3


enaMARM
MARMuxOut
16
Buss
16
enaPC
PC
16
Buss
16
DR
3
SR1
3
SR2
3
regWE
Reg File
Buss
16
IR
8
MARMux
selPC
2
PC
selMAR
eabOut
16
ldPC
clk
reset
clk
reset
Memory
ldMAR
ldMDR
memWE
selMDR
clk
reset
selEAB1
selEAB2
2
EAB
PC
16
Rb
16
Ra
16
IR
11
Ra
16
IR
6
N
Z
P
aluControl
2
ALU
ldIR
clk
reset
IR
16
IR
NZP
flagWE
aluOut
16
mdrOut
16
enaALU
Buss
16
Buss
16
enaMDR


LC3
Controller
IR
N,Z,P
Control
Signals
LC3
Datapath
Programmed
Memory