

# SingleCycleImplementation

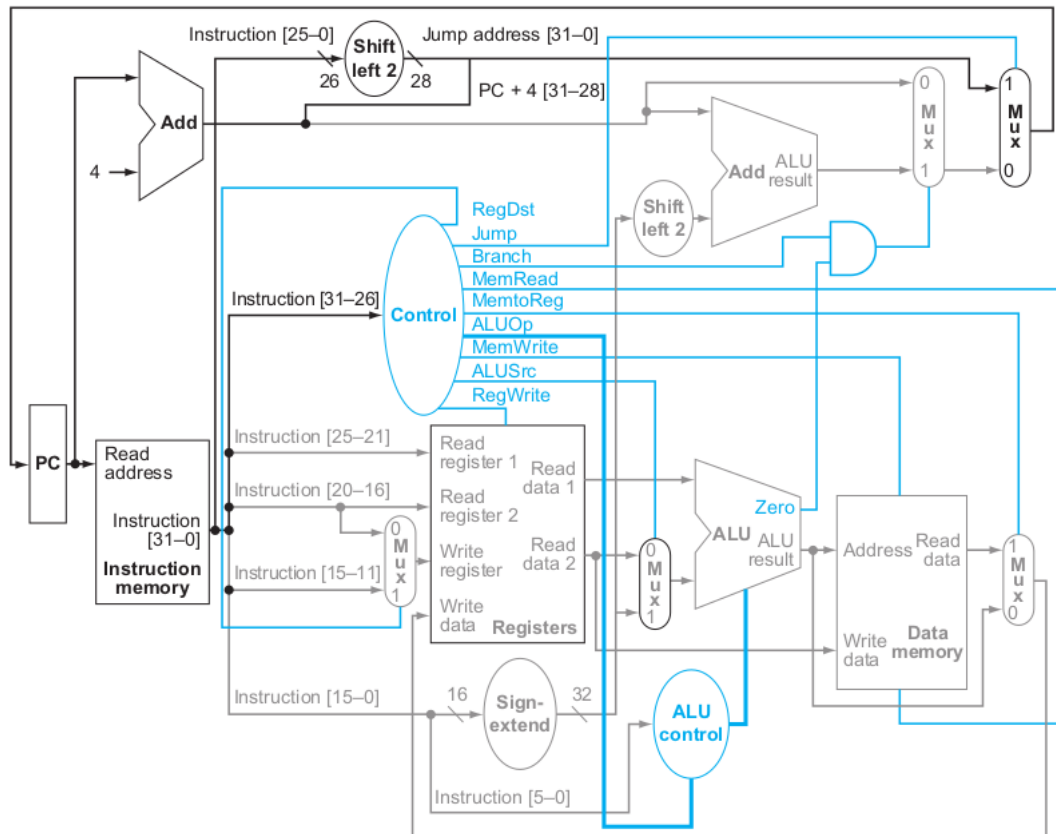


FIGURE 4.24 The simple control and datapath are extended to handle the jump instruction.

- Single Cycle
- No pipelining
- Not Synthesizable
- Supports add, sub, sw, ld, beq and b.

# PipelinedImplementation1

## Project 1

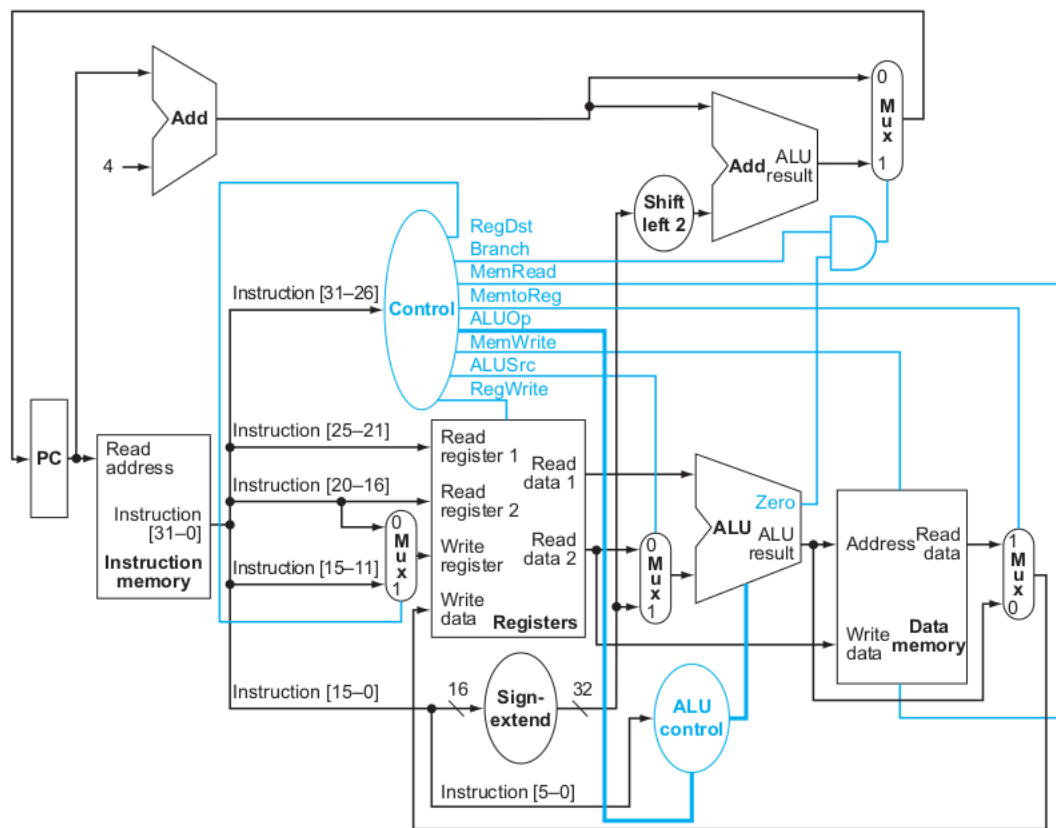


FIGURE 4.17 The simple datapath with the control unit.

- Single Cycle
- Simple Pipelined
- Synthesizable
- Supports add, sub, sw, ld and beq.
- No Hazard Mangement.

## Project 2

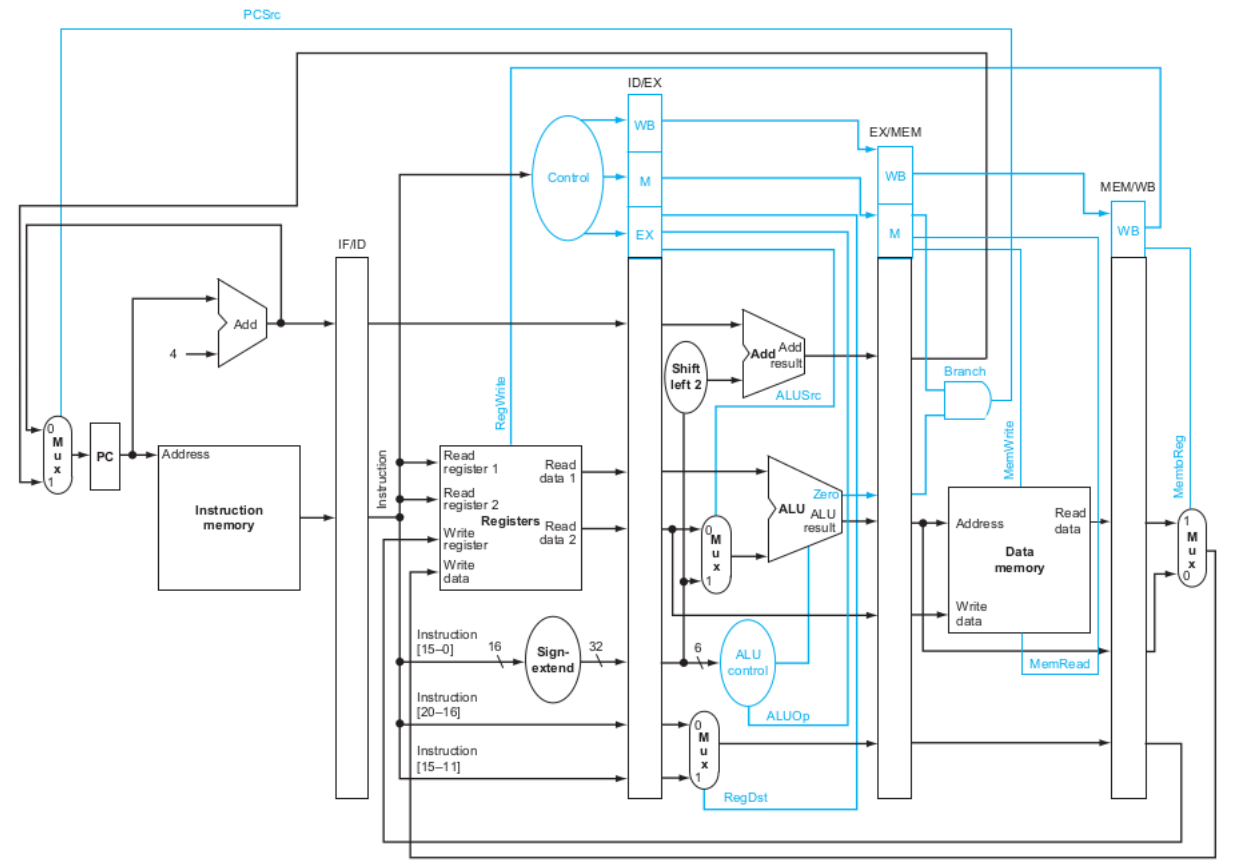


FIGURE 4.51 The pipelined datapath of Figure 4.46, with the control signals connected to the control portions of the pipeline registers.

- Single Cycle
- Simple Pipelined (same as Project 1 but with separate blocks)
- Synthesizable
- Supports `add`, `sub`, `sw`, `ld` and `beq`.
- No Hazard Mangement.

## PipelinedImplementation2

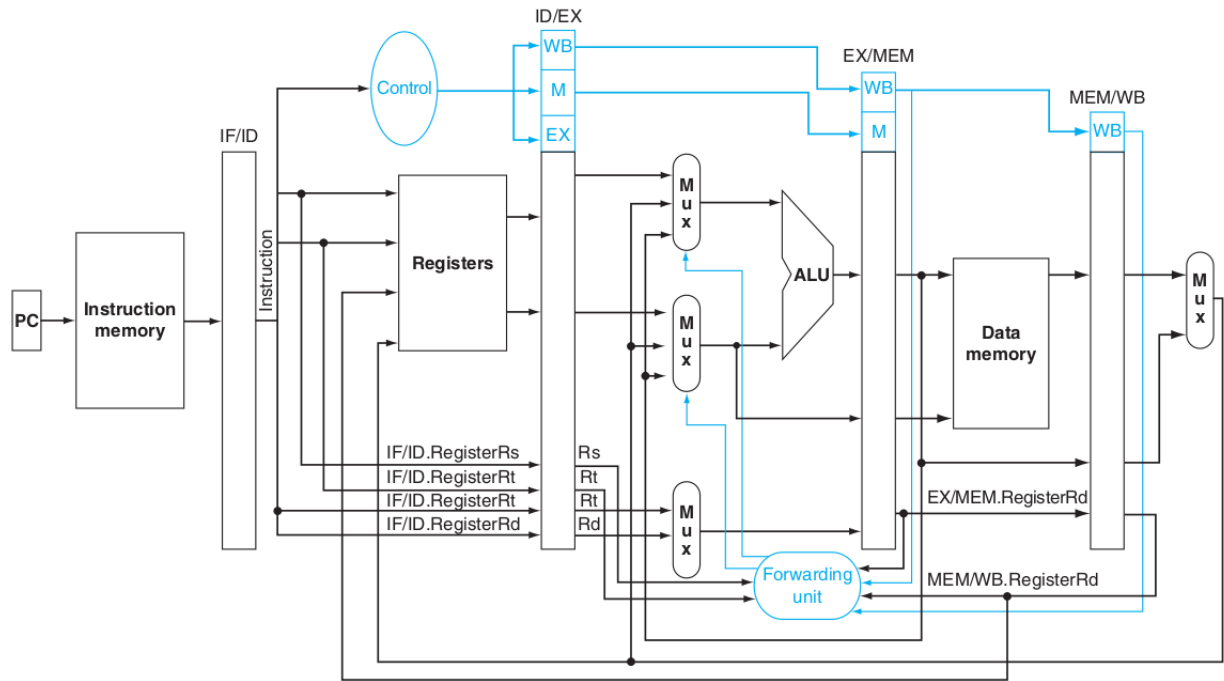


FIGURE 4.56 The datapath modified to resolve hazards via forwarding.

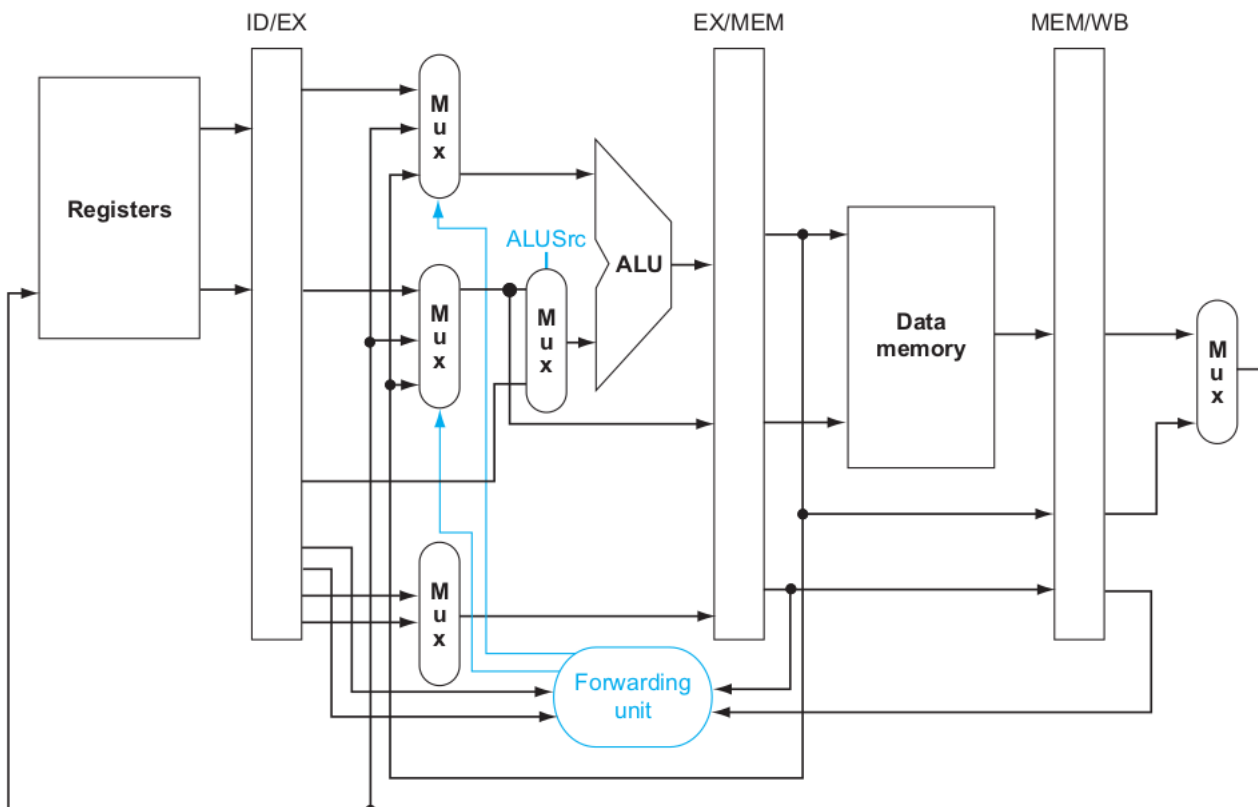


FIGURE 4.57 A close-up of the datapath in Figure 4.54 shows a 2:1 multiplexor, which has been added to select the signed immediate as an ALU input.

- Single Cycle
- Pipelined with Forwarding for Data Hazard
- Synthesizable

- Supports add, sub, sw, ld and beq.
- Data Hazard between stages (see below).

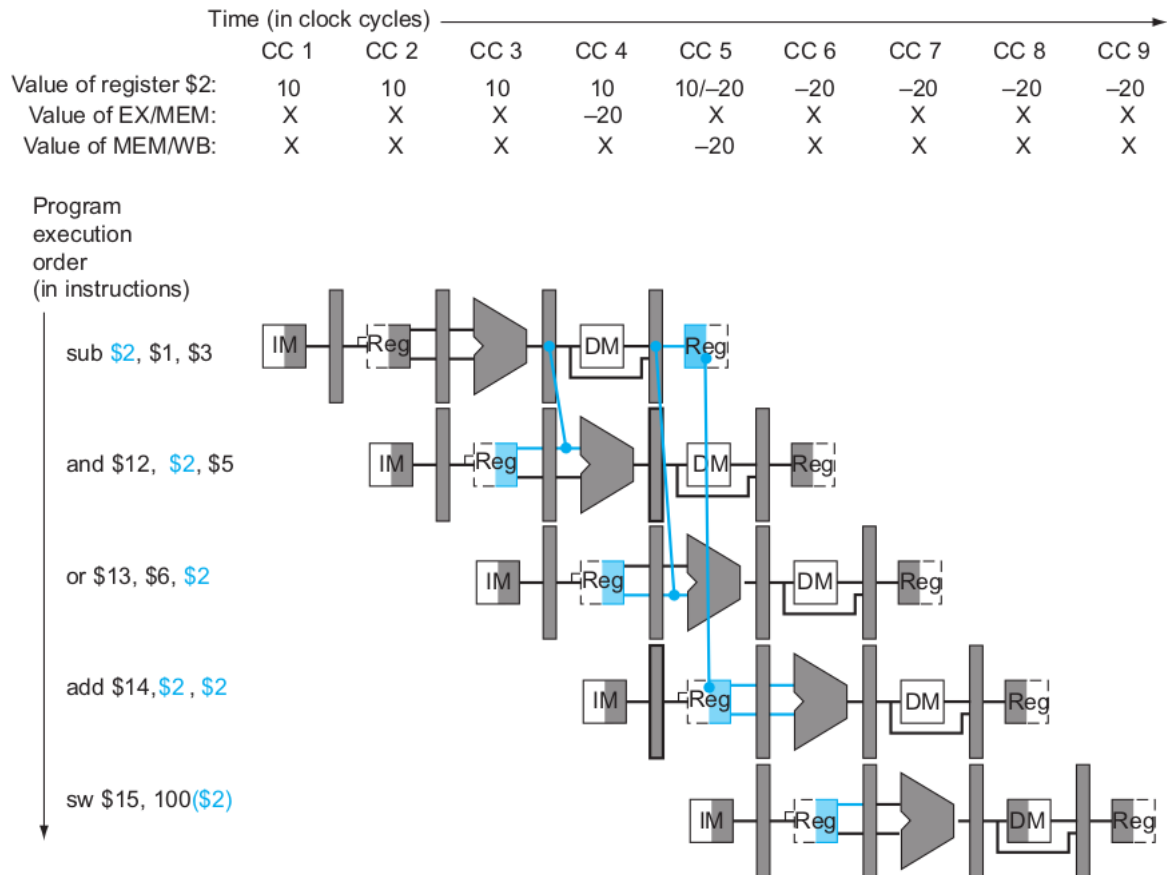
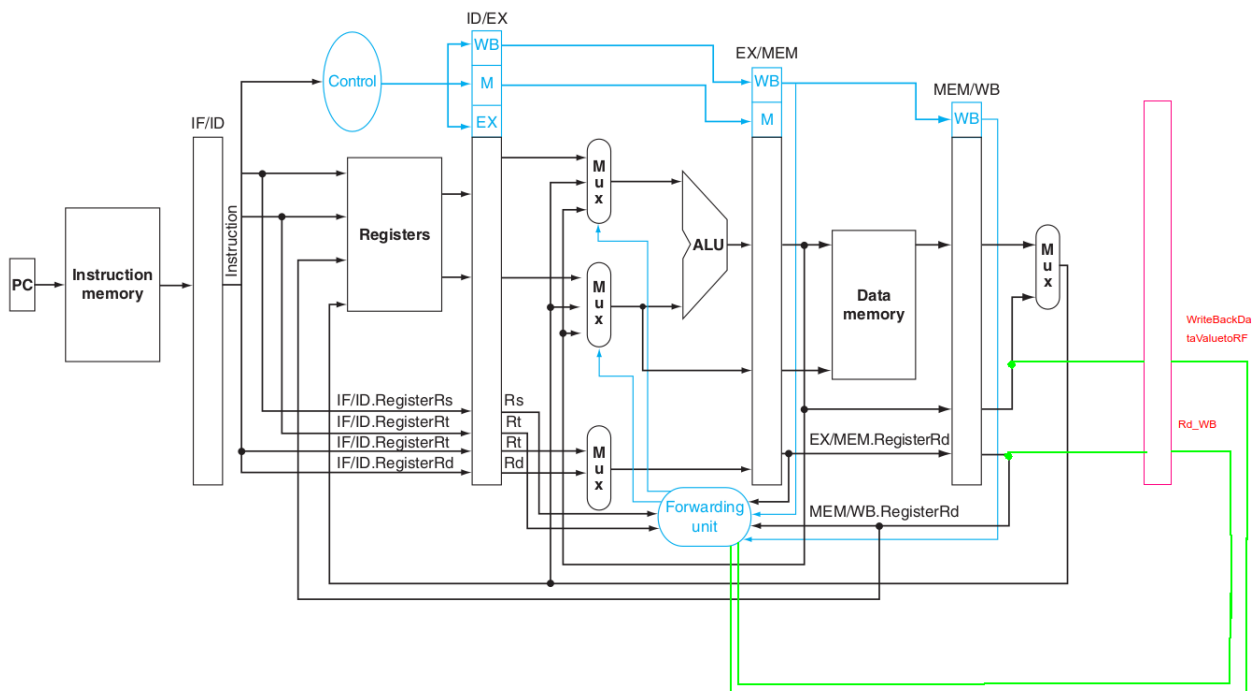


FIGURE 4.53 The dependences between the pipeline registers move forward in time, so it is possible to supply the inputs to the ALU needed by the AND instruction and OR instruction by forwarding the results found in the pipeline registers

- Also, added another forwarding(see below) from writeback to execution.

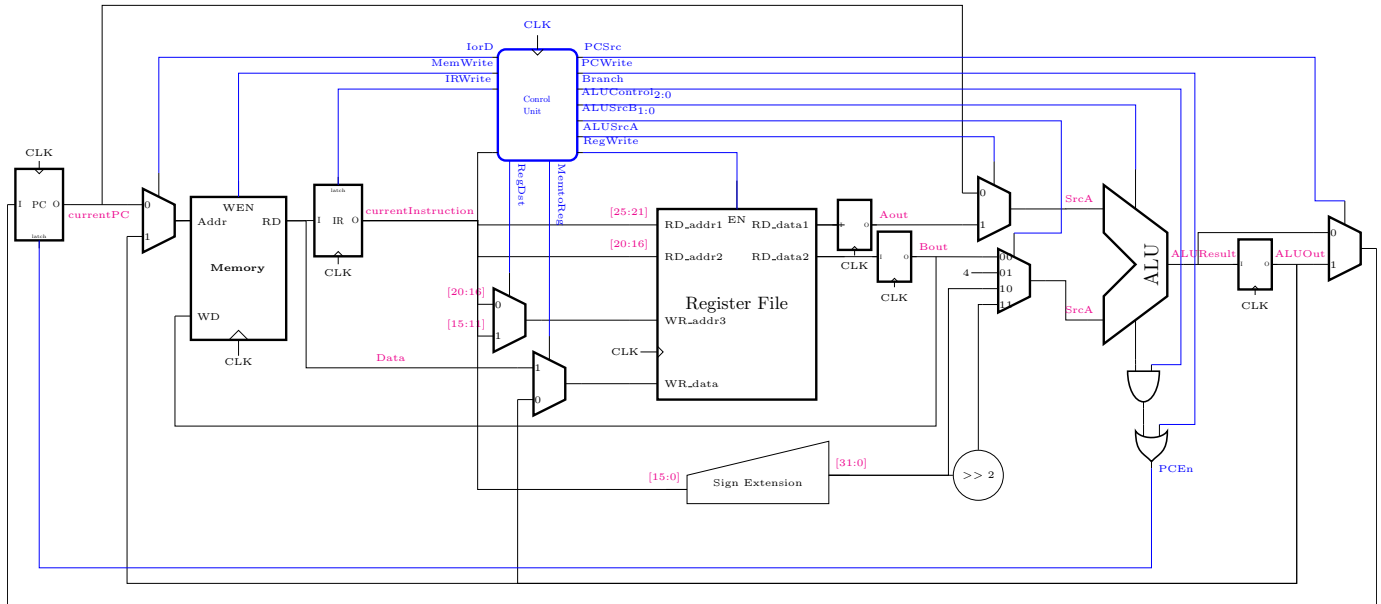


## Note

Left till forwarding in **Computer Organization and Design MIPS Edition The HardwareSoftware Interface** (The Morgan Kaufmann Series in Computer Architecture and Design) by David A. Patterson, John L. Hennessy (z-lib.org).pdf - Haven't implemented stalls - Pg no. 313.

## MultiCycleImplementation

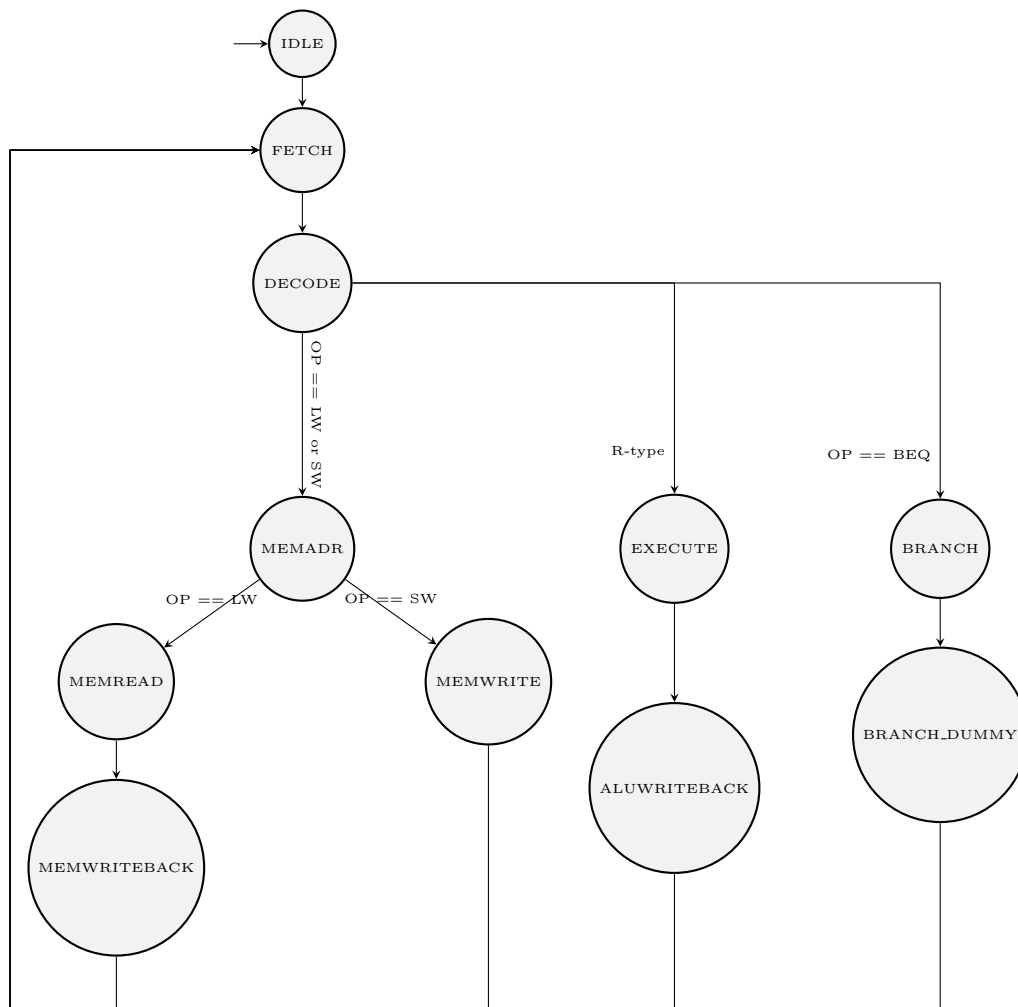
### Project 1



Removed the Data register as already data's are registered from memory

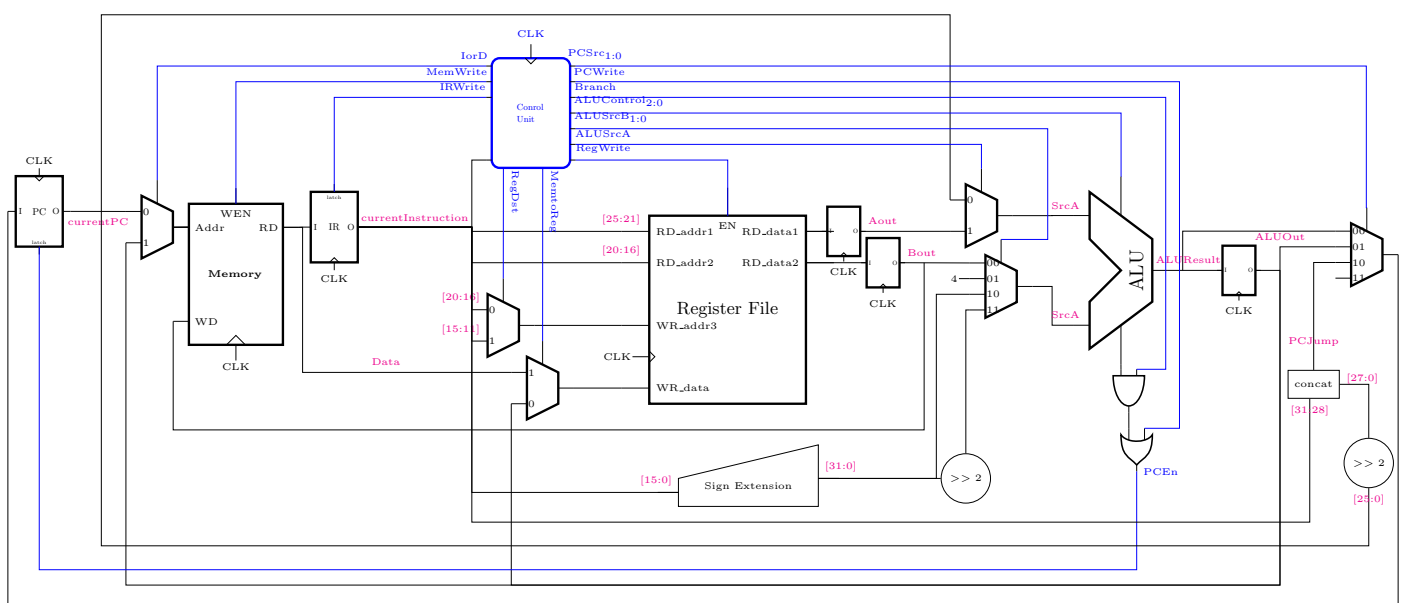
- Multi Cycle
- No data Hazard
- No Control Hazard
- Synthesizable
- Supports add, sub, sw, ld and beq.
- Uses FSM
- Shrunk from Single Cycle

## State Diagram



Added a new dummy state for —beq—

## Project 2

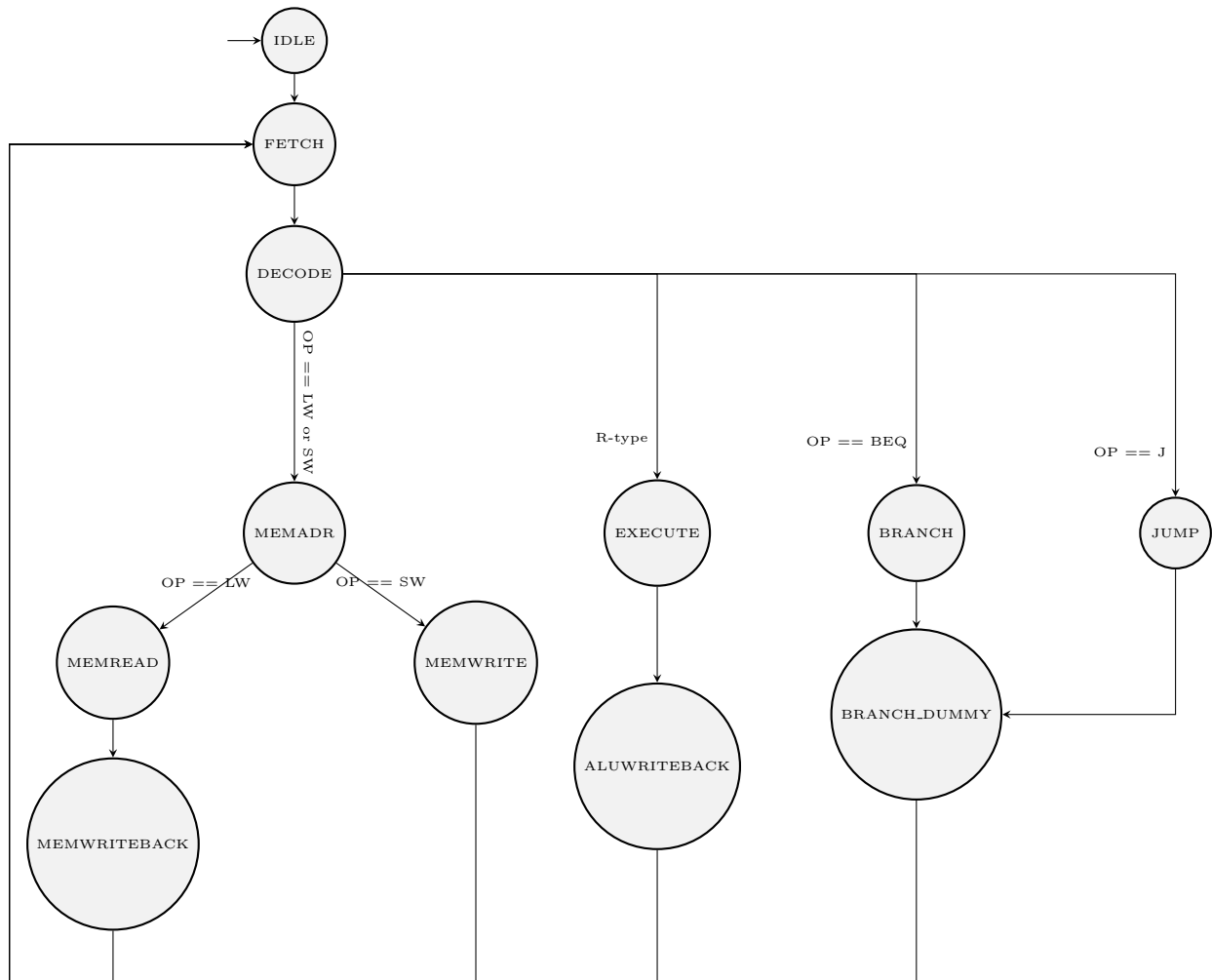


added Support for Jump instruction

- Multi Cycle

- No data Hazard
- No Control Hazard
- Synthesizable
- Supports add, sub, sw, ld, beq and j.
- Uses FSM
- Shrunked from Single Cycle

### State Diagram



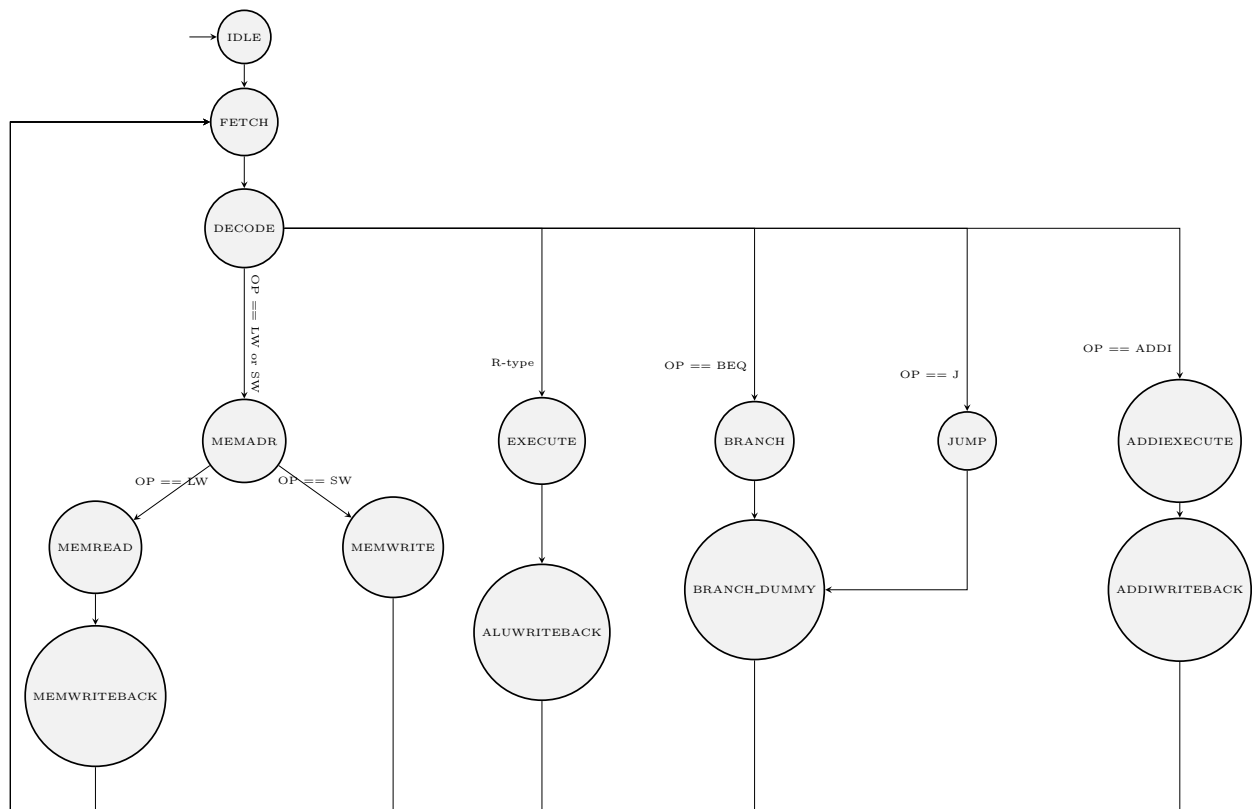
Added a new dummy state for —j—

### Project 3

- Multi Cycle
- No data Hazard
- No Control Hazard
- Synthesizable
- Supports add, sub, sw, ld, beq, j and addi.
- Uses FSM
- Shrunked from Single Cycle



## State Diagram



Added a new dummy state for —addi—

## References

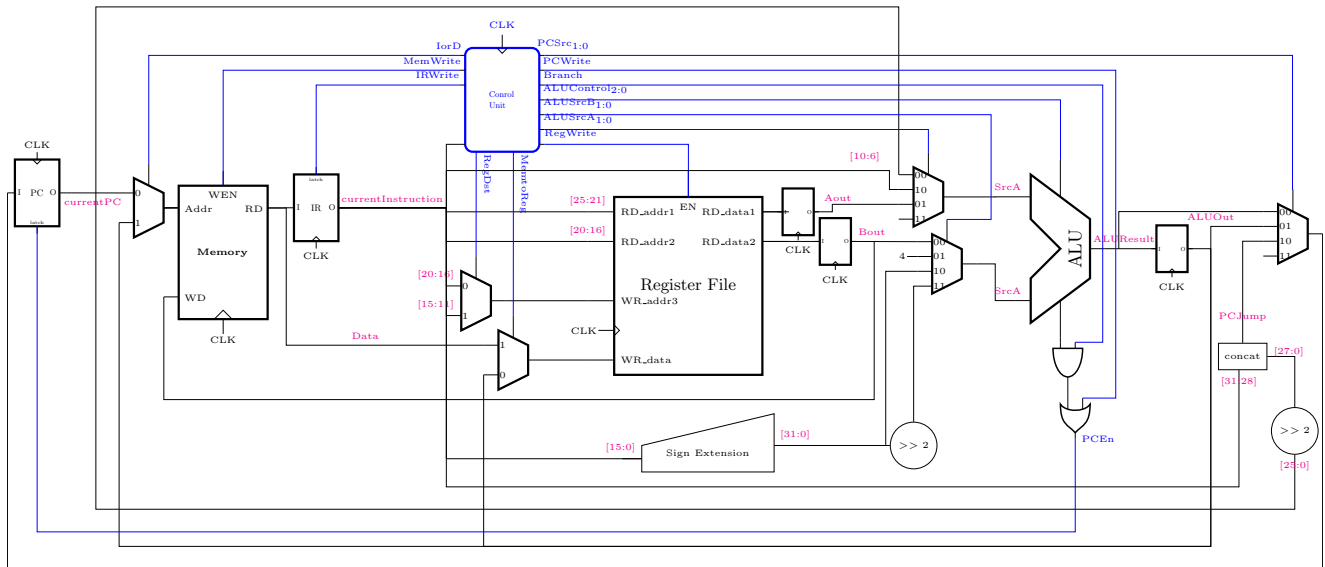
- L1
- L2
- L3

## MultiCycleImplementation\_NewOnes

### Project 1

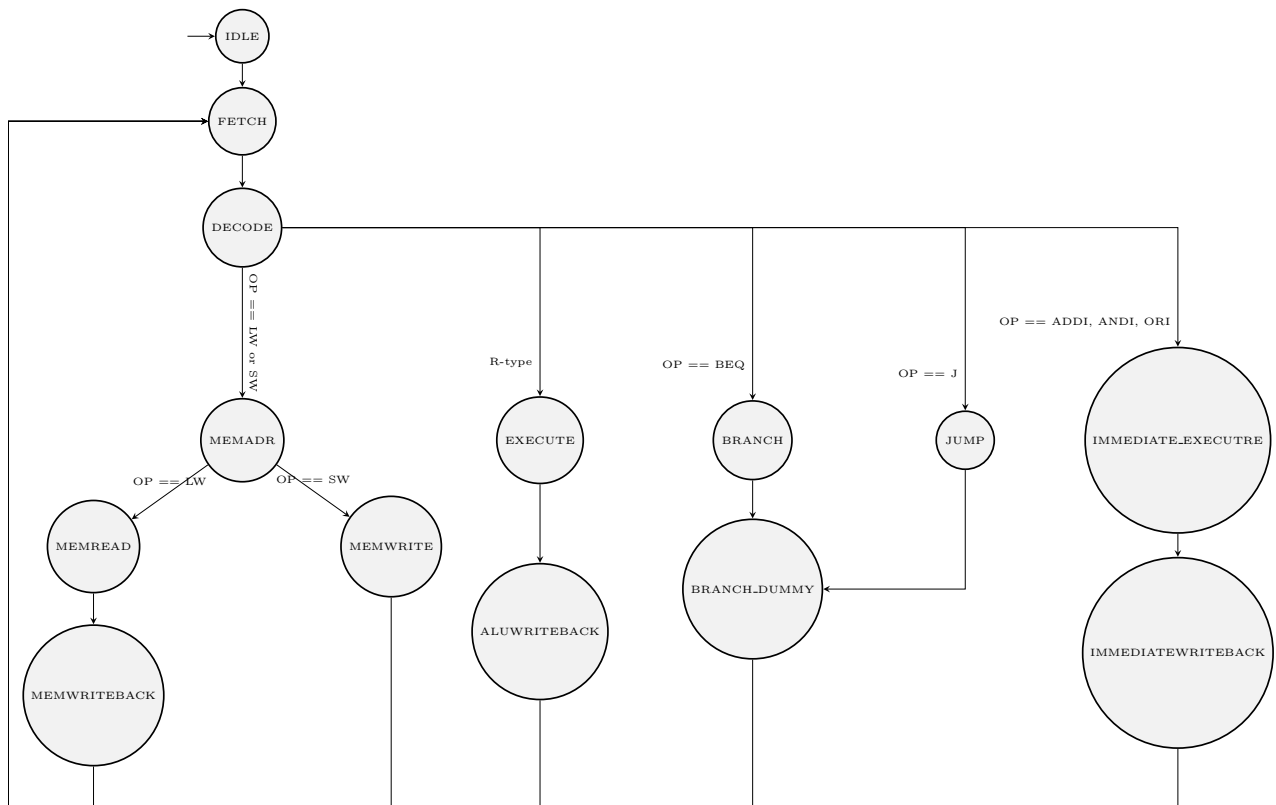
- Added support for "and, or, nor".

## Project 2



Changed ALUSrc A Mux

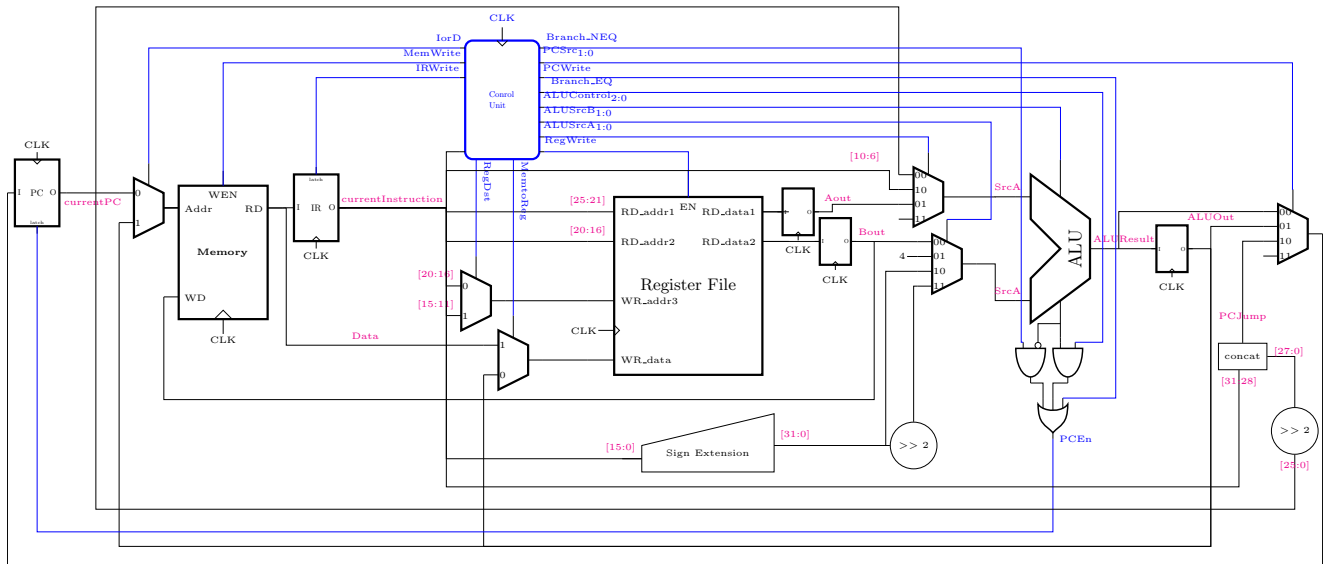
## State Diagram



Added support for addi, and ori,sll, srl

- Modified controller FSM add immediate to general immediate.
- Modified DataPath to have shift based operation (ALU is included with shift operations with operand reversed).
- Replaced 2x1 mux of ALUSrcA with 4x1 mux with the third input being the 5bit value of immediate.
- Modified controller FSM to incorporate shift operation with R type instruction.
- Added support for "addi, andi, ori, sll and srl".

## Project 3



Added support for bneq

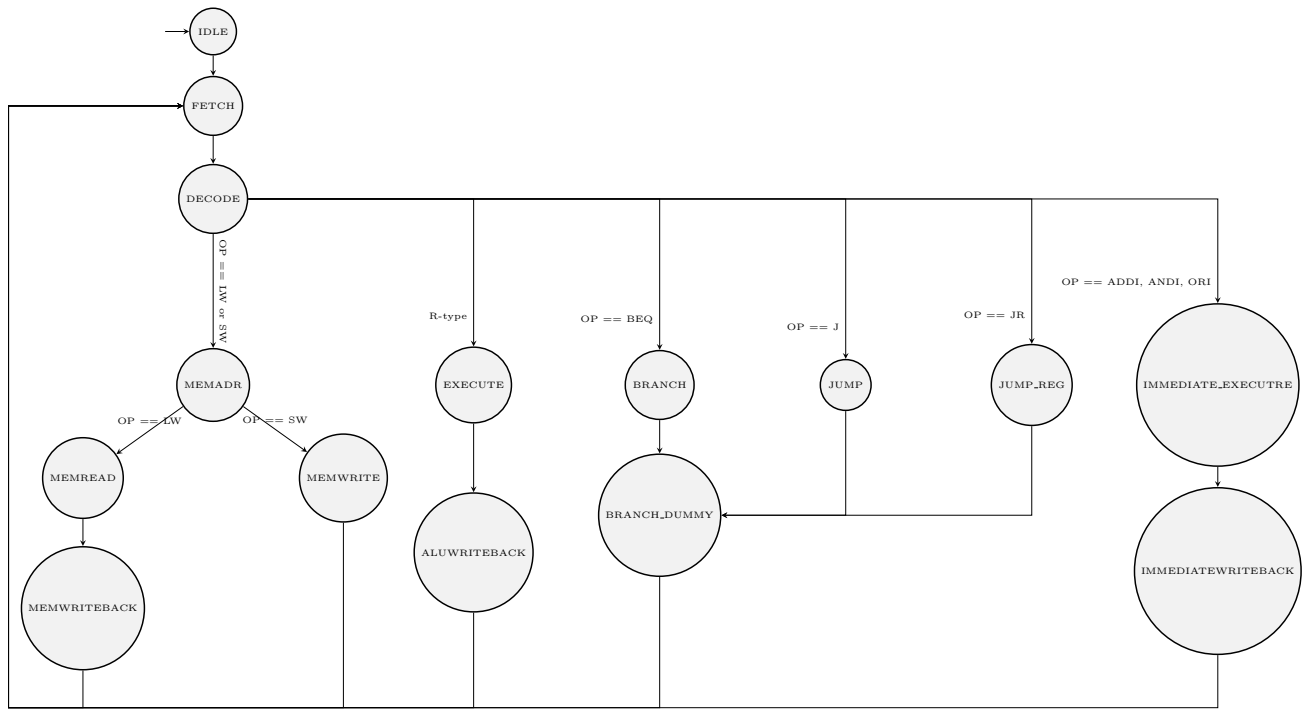
- Modified Data Path to incorporate "bne" by adding Branch\_neq signal from controller and AND with  $\overline{isZero}$  which is ORred with PCWrite.
- Modified FSM to have both "beq" and "bne" in same BRANCH state.

## MultiCycleImplementation\_NewOnes2

### Project 1

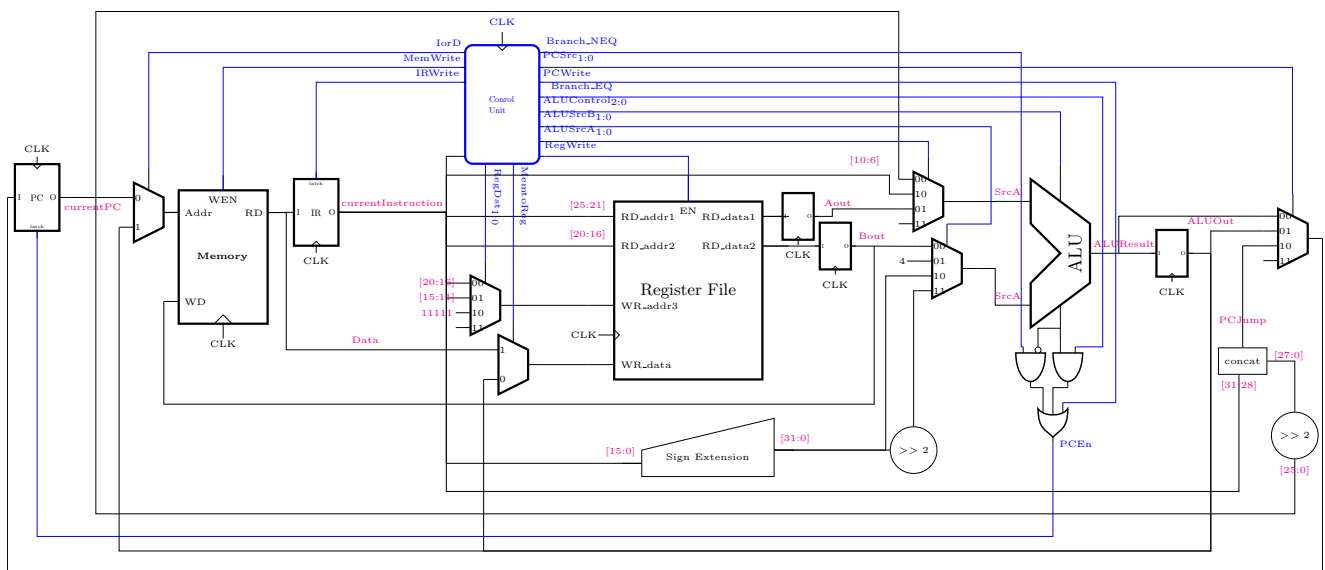
- Added support for "jr"
- By modifying FSM to uses a state where one operator is the register rs, and other operator is 0 (see instructions) and code.
- The address must be in 4 aligned.
- Note: Also jr will jump to address +4 value.

## State Diagram



Added support for jr

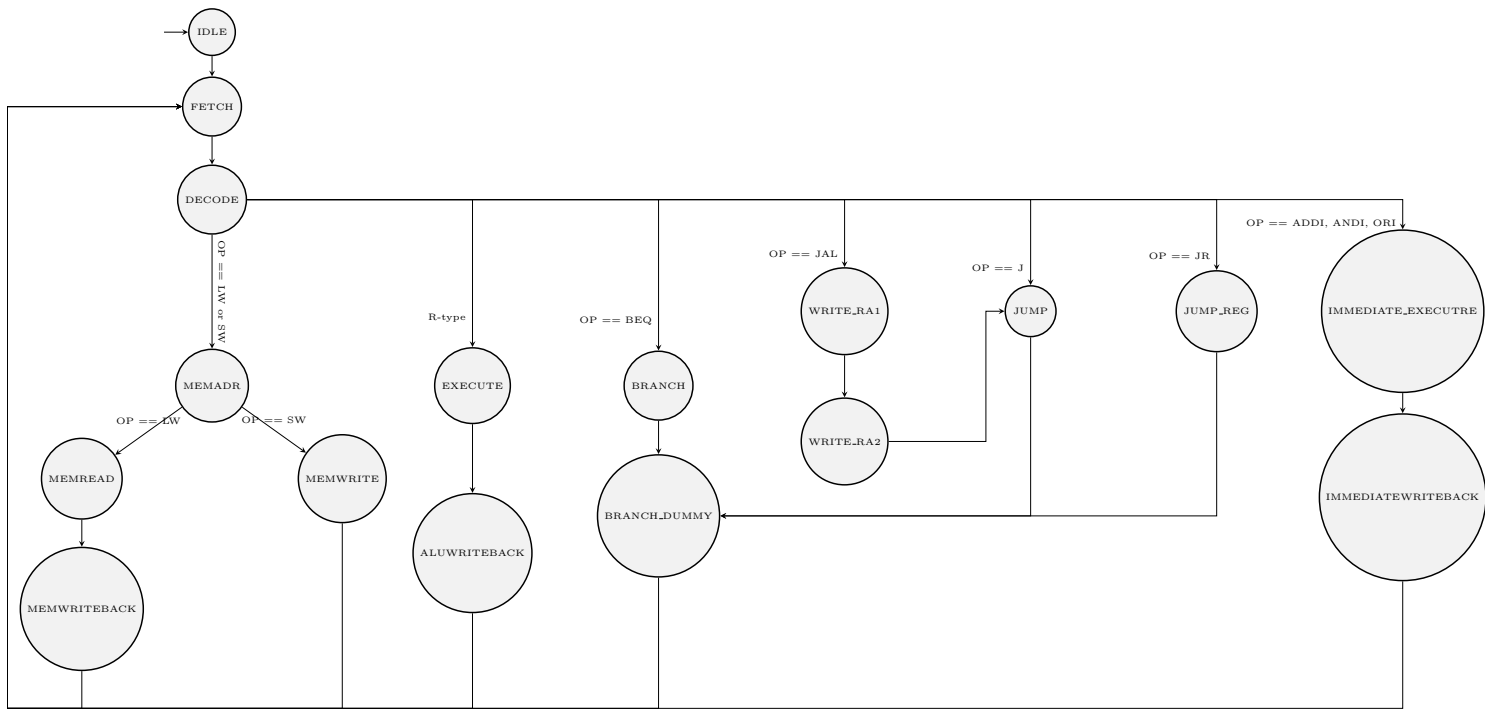
## Project 2



Added support for jal

- Added support for "jal" – stored pc+4 in \$ra and jump to the location.
- Modified FSM to have extra states for jal
- Modified DataPath by changing the write register mux to 4x1.

## State Diagram



Added support for jal