# Lecture 9, Exceptions and Beyond

## Exceptions

- emergencies requiring change in normal flow of instruction execution
- exception: unexpected event from within processor e.g. arithmetic overflow
- interrupt: unexpected event from external environment e.g. input / output request
- need to
  - find out what happens and deal with accordingly
  - resume normal execution after the emergency

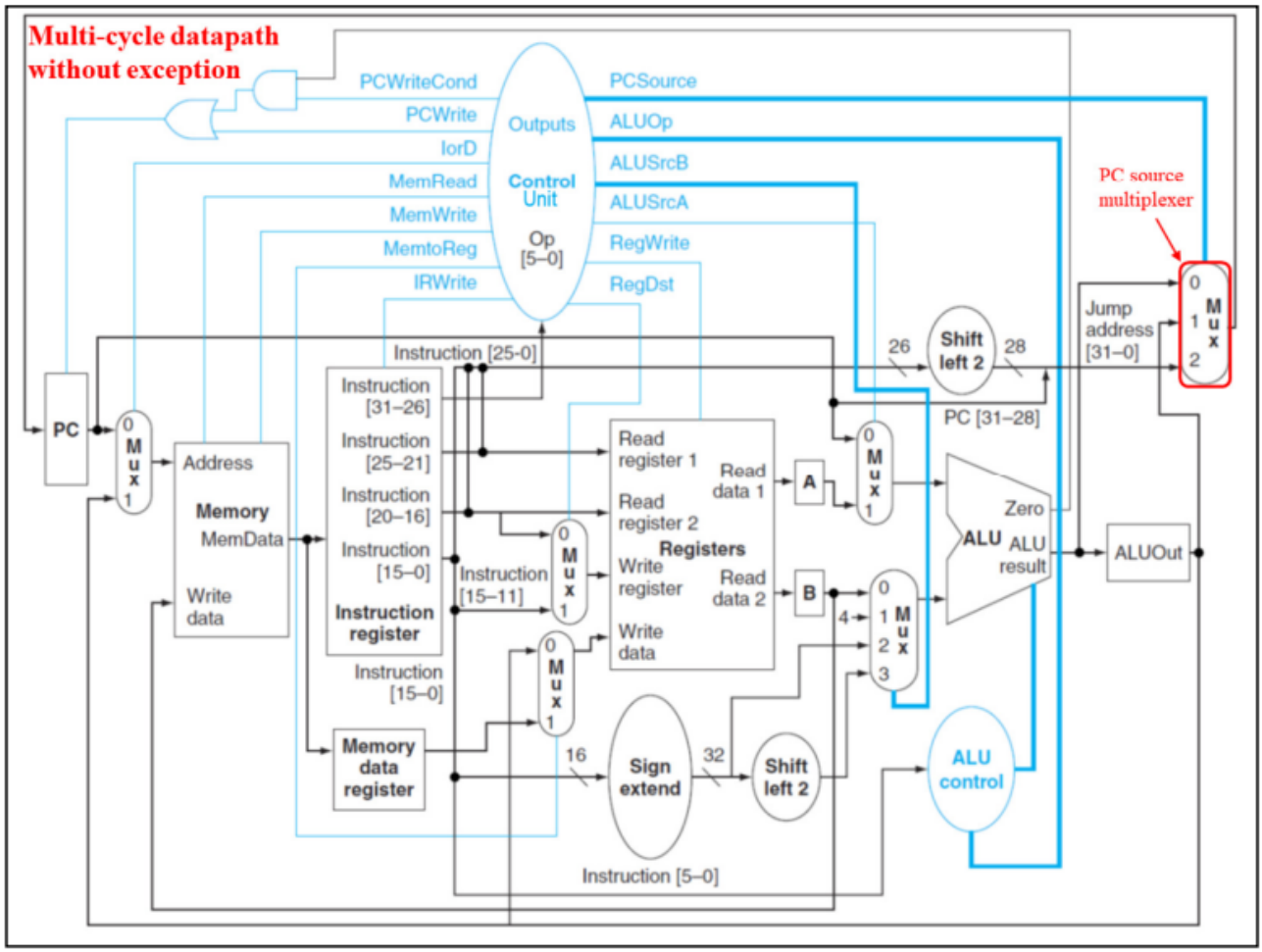
To deal with an unexpected event, the key is to stop the normal execution, find out what happens, deal with it, and then resume the normal execution.

## Handling Exceptions

- save indication of the cause of exception in the cause register
- new control signal IntCause from control unit to cause register
- e.g. IntCause = 0 for unknown opcode
  = 1 for arithmetic overflow
- save restart address in exception PC (EPC)
- transfer control to operating system by executing instruction at exception address
- code at exception address can find out what happens by inspecting cause register

wl 2

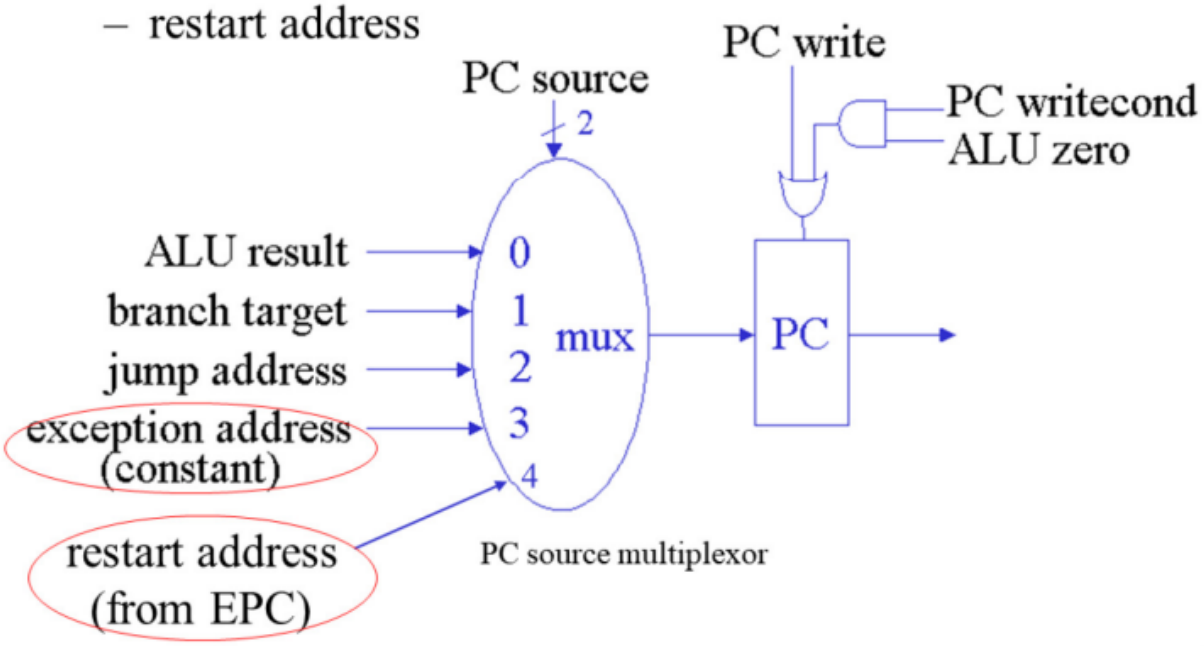
There are two important addresses in handling exceptions. The address of the instruction which causes the exception is called the restart address and is stored in the EPC register. The address that contains the instruction to deal with exceptions is stored at the exception address in memory.




This is the multi-cycle datapath without support for exceptions.

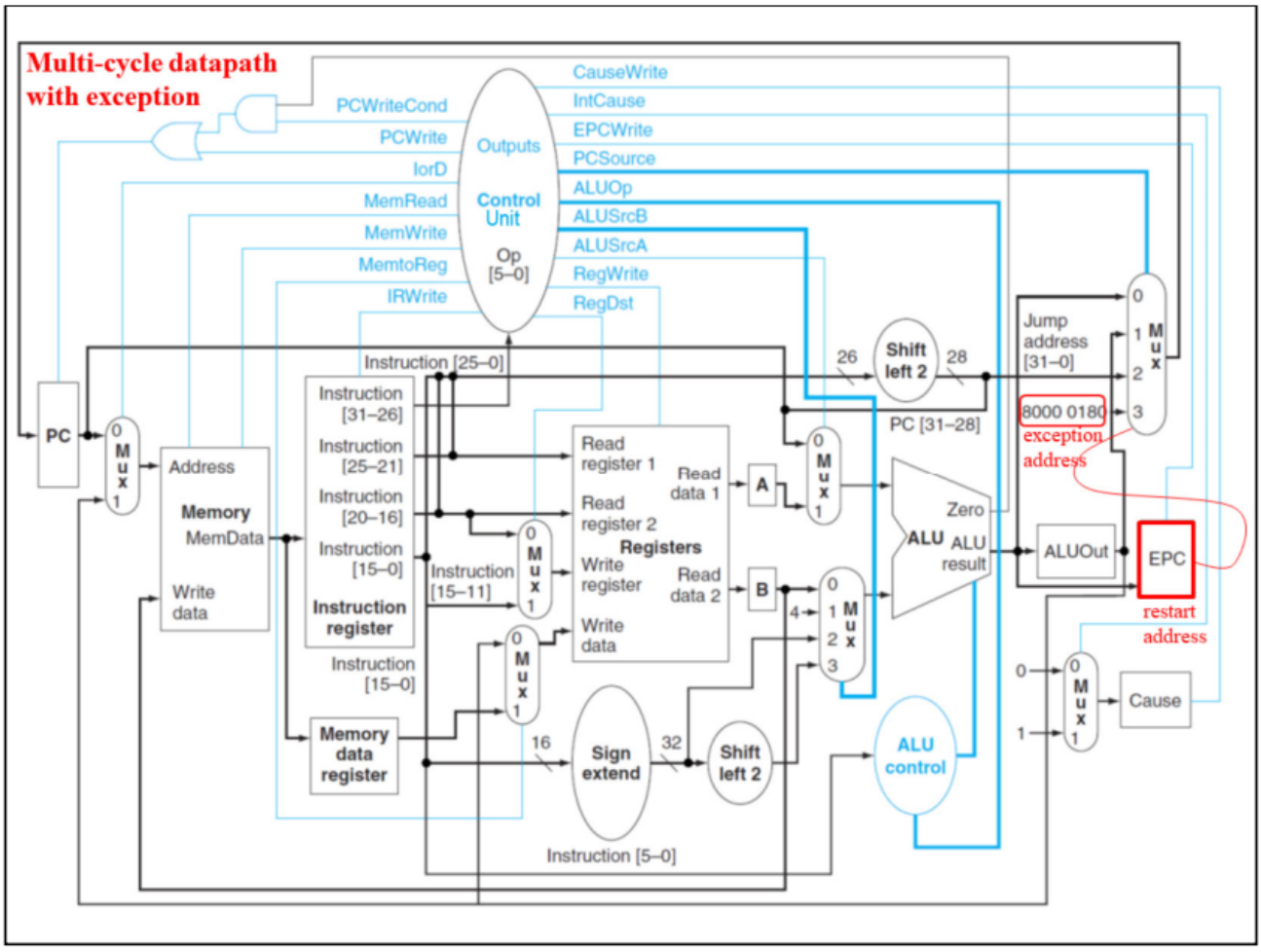
# Datapath Modification

- add to PC source multiplexor
  - exception address
  - restart address



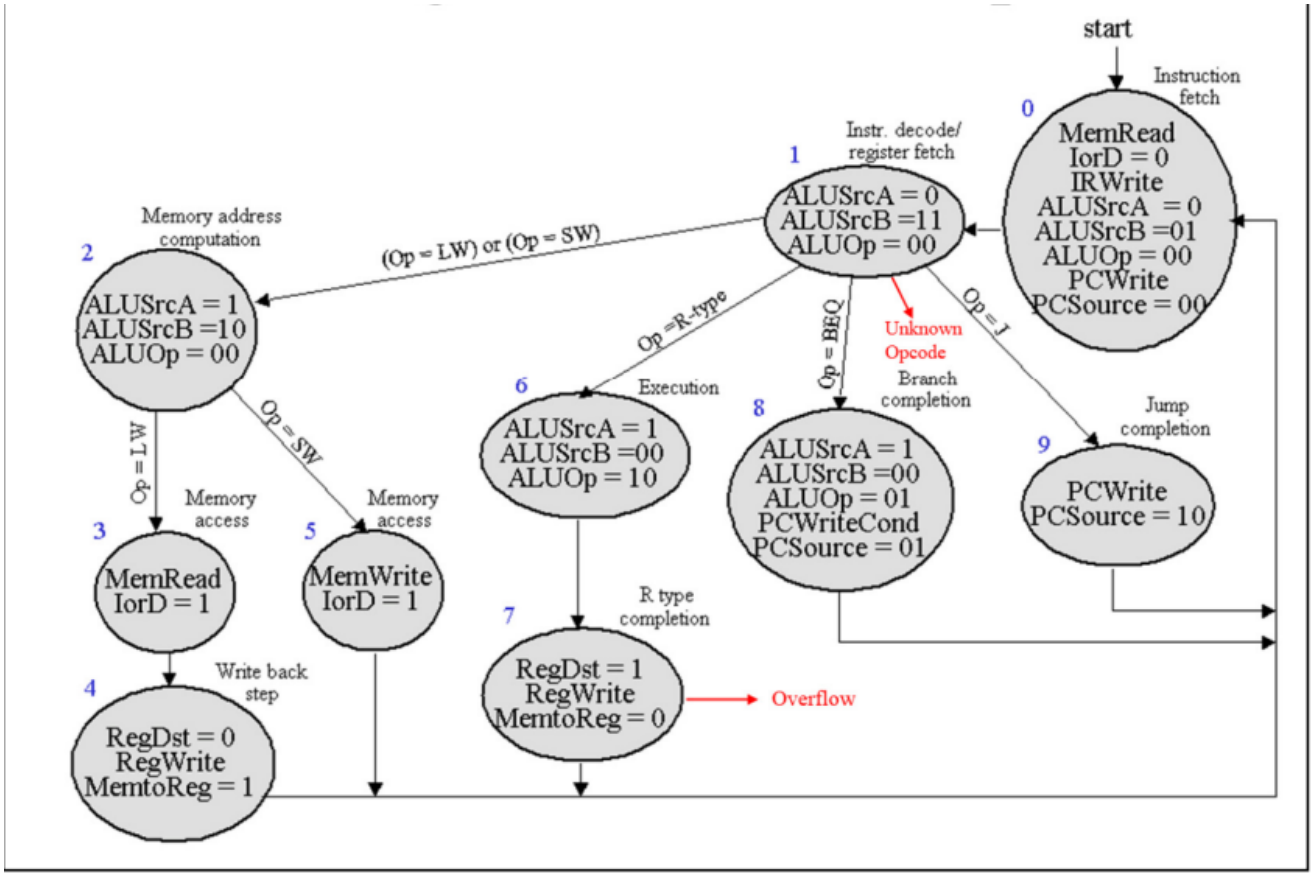

wl 2024

The main change to the datapath is to enlarge the PC source multiplexor so that the exception address and the restart address can be selected to be sent to the PC under the right conditions.



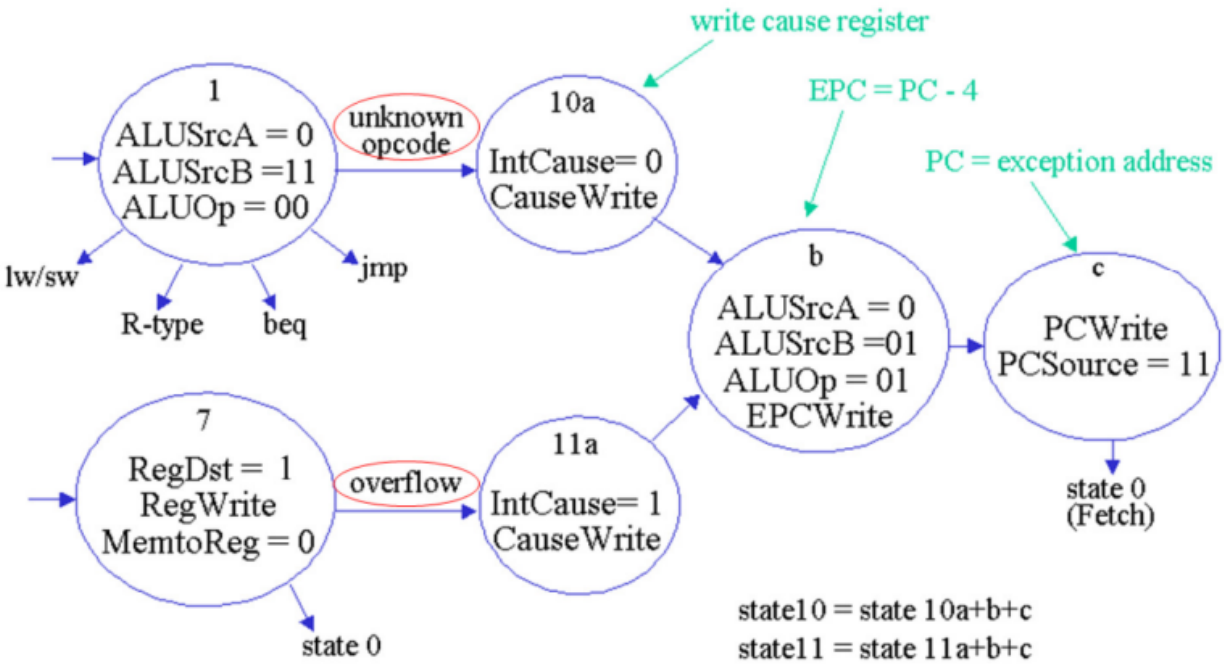
This is the multi-cycle datapath with support for exceptions. The exception address is the constant value 8000 0180.

# State Diagram Without Exceptions

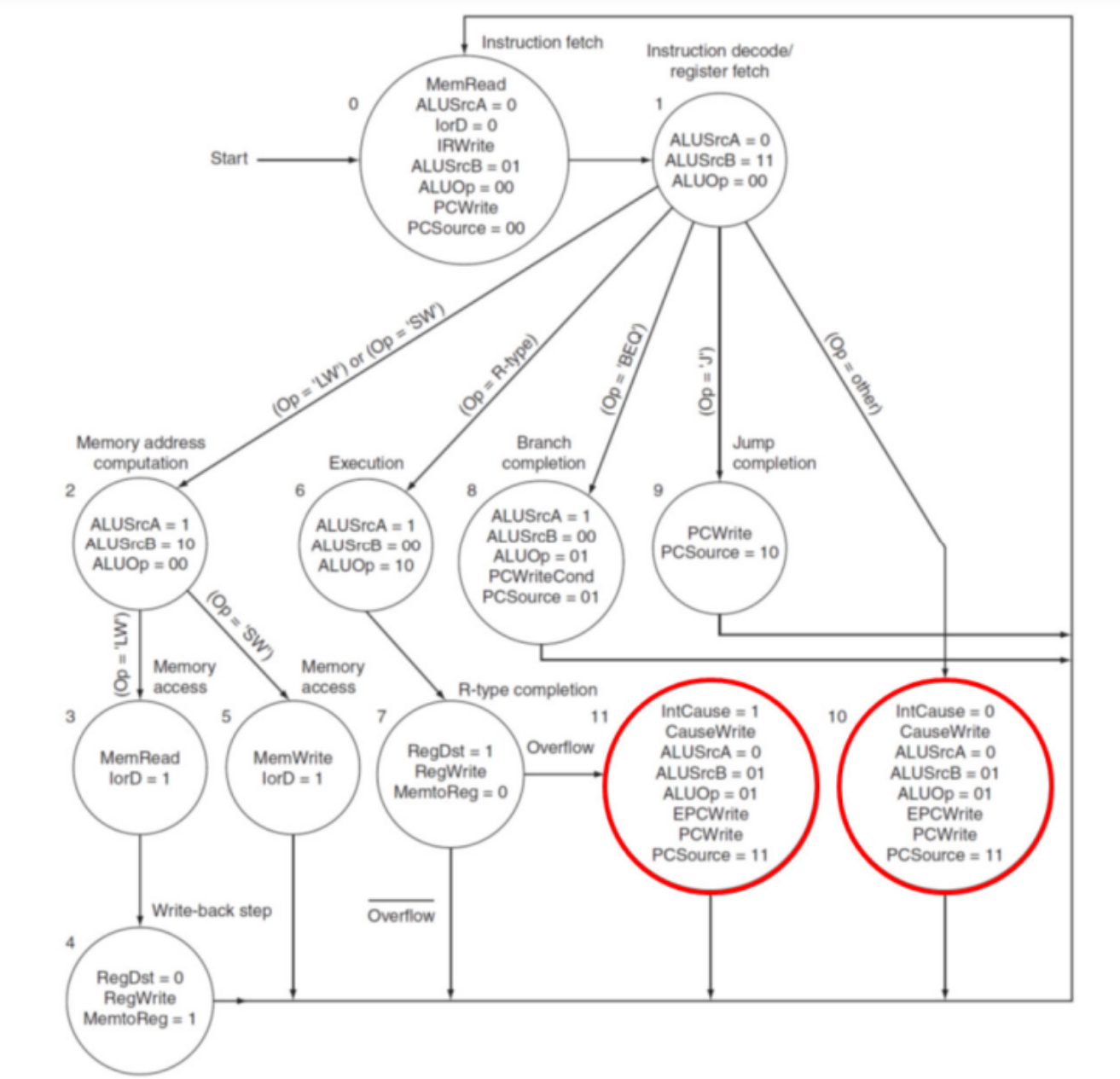



To support handing of exceptions to address Unknown Opcode and ALU Overflow, we need to modify the state diagram for the Control Unit.

# Modified State Diagram




The modifications are simple. Indeed the only difference in supporting different exceptions is to store the appropriate value for IntCause in the Cause Register; the rest are identical. The step $EPC = PC - 4$ is to store the instruction that causes the exception into the EPC; we need the $PC - 4$ value since the PC has already been incremented to point to the next instruction.




The state diagram for the Control Unit to support the two exceptions.