Fetch Unit Operation

CSE 141L Lab 2

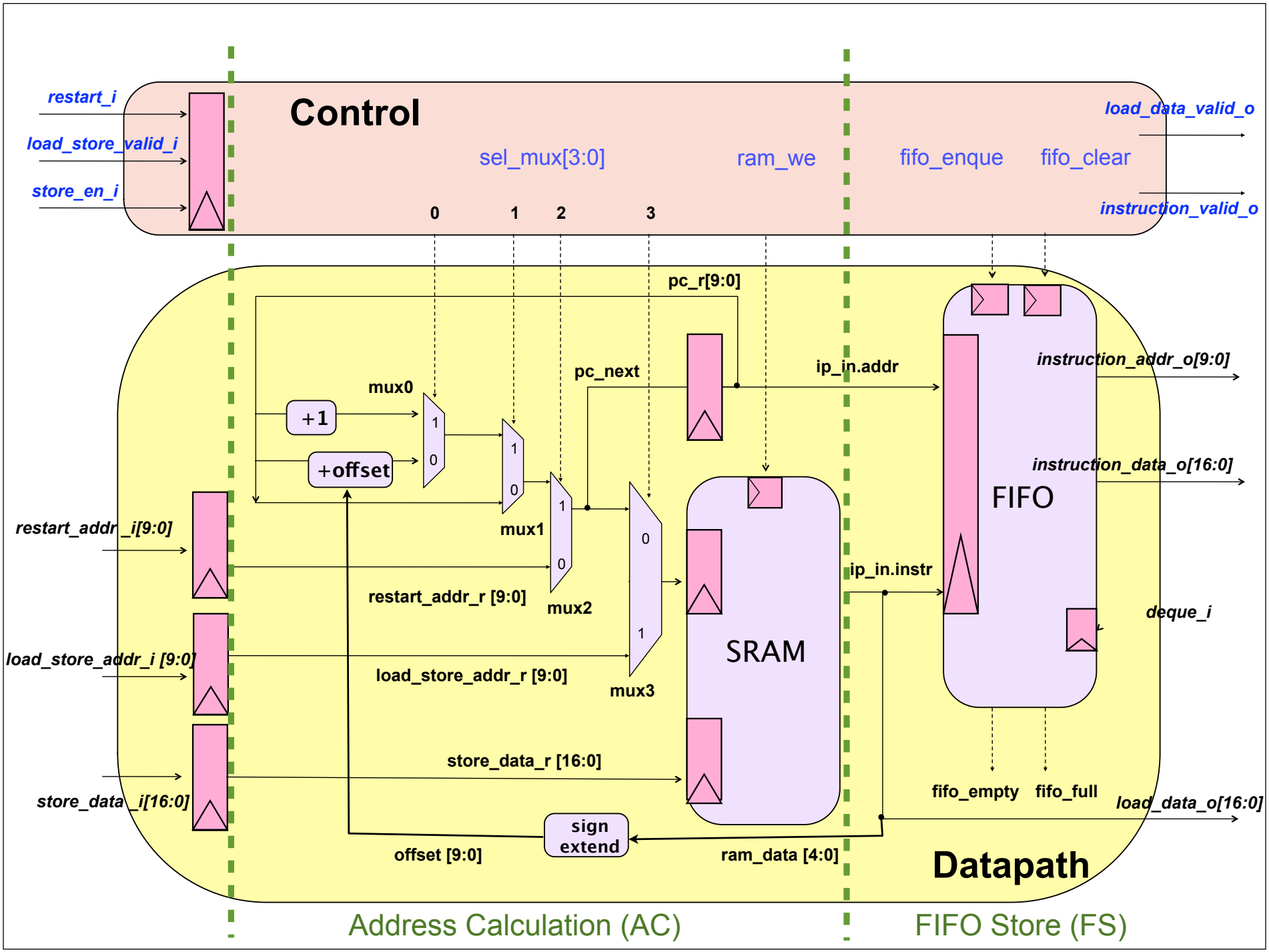
Front-End vs Back-End

- Front-end *fetches* instructions (Lab 2)
  - Reads from instruction memory
  - Stores fetched instructions in FIFO
  - Predicts which addresses to fetch
- Back-end *executes* instructions (Lab 3)
  - Pulls instructions from front-end's FIFO
  - Reads/writes to data memory
  - Must check correctness of front-end prediction.

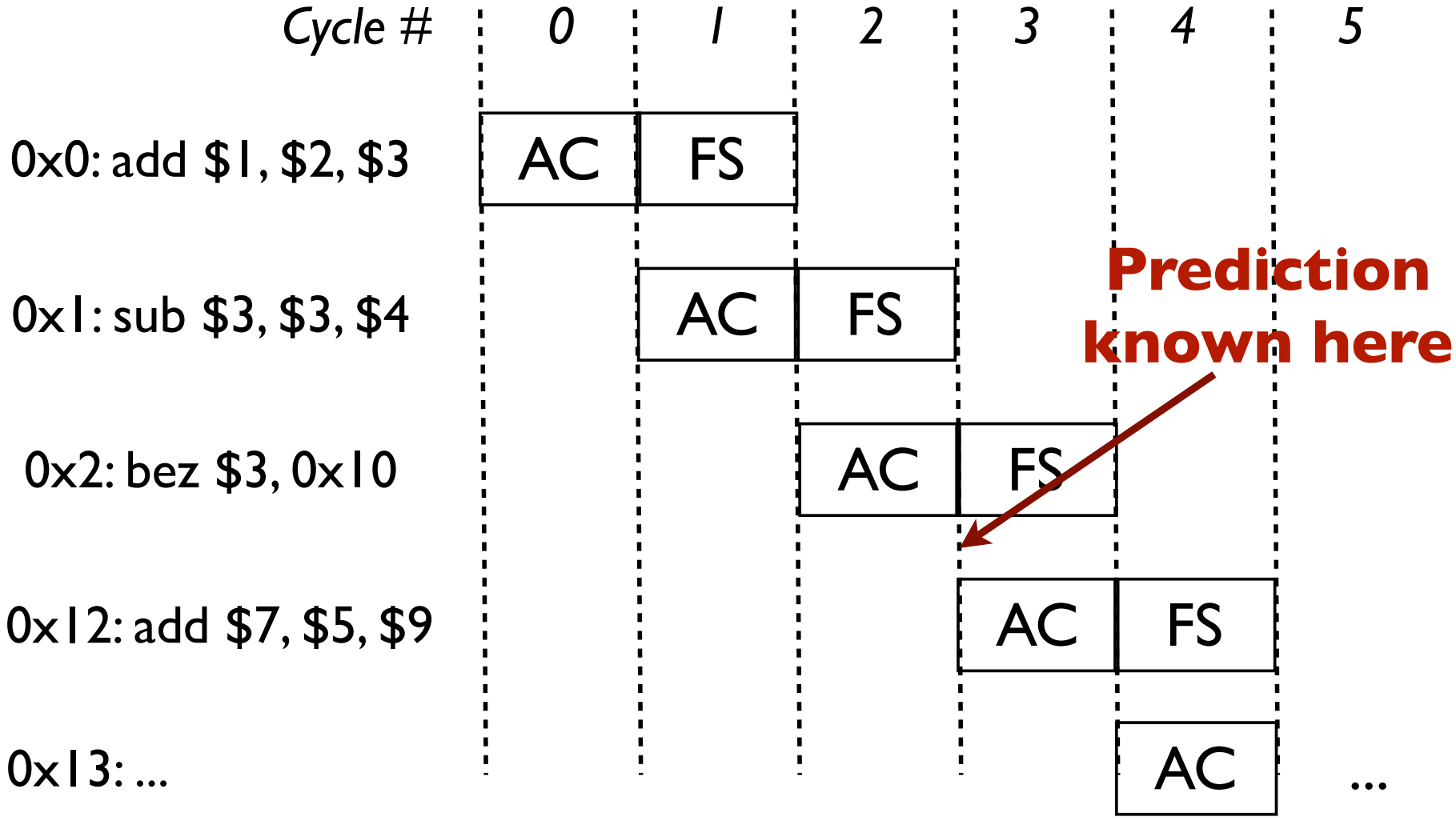
Why separate fetch from execution?

Pipelined Fetch Unit

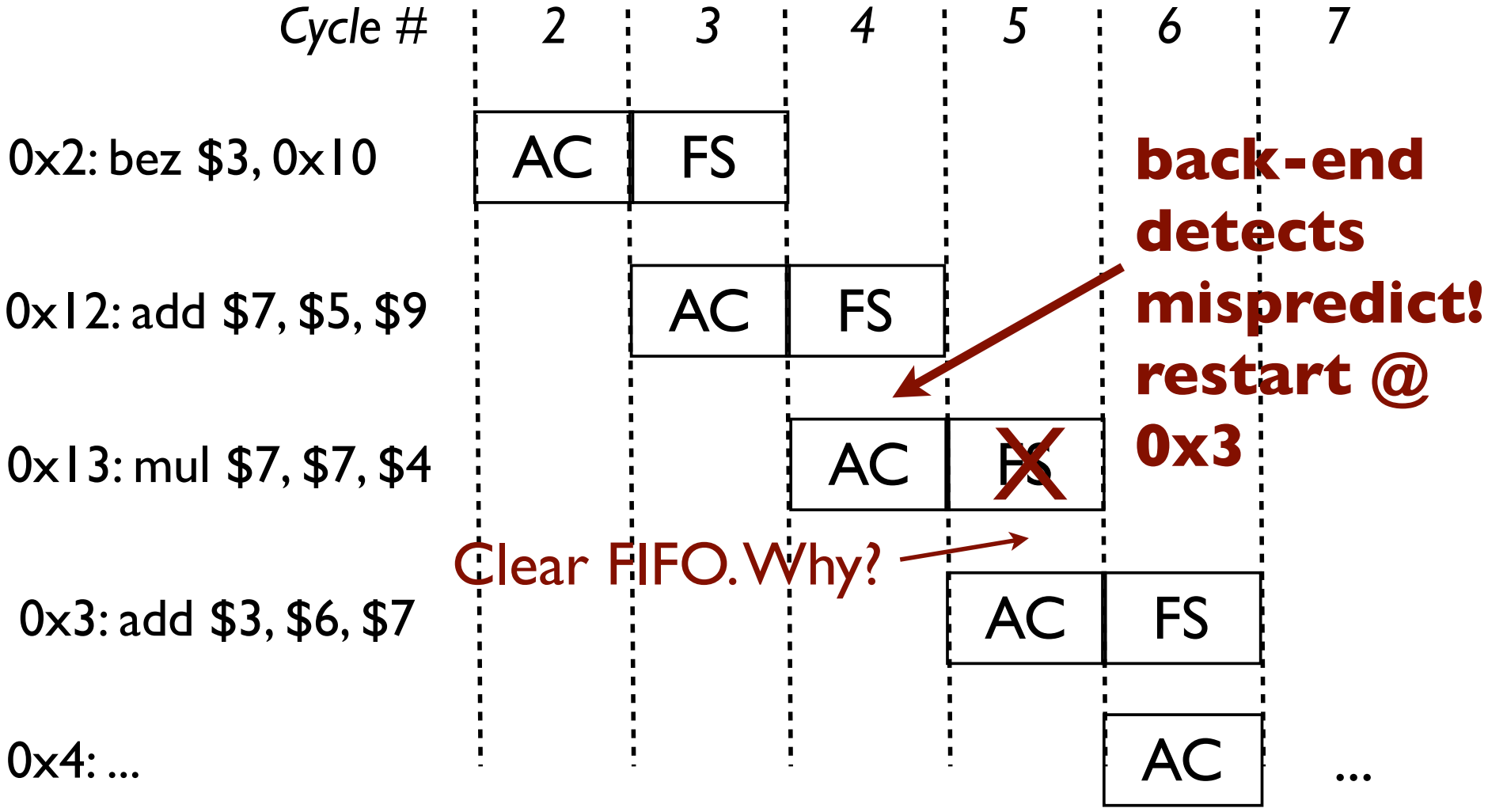
- Fetch unit split into two separate stages
  - Address Calculation
    - Determines what address goes into SRAM
  - FIFO storage
    - Stores instruction address and SRAM output into FIFO


Control
restart_i
load_store_valid_i
store_en_i
sel_mux[3:0]
ram_we
fifo_enque
fifo_clear
load_data_valid_o
instruction_valid_o
0
1
2
3
pc_r[9:0]
pc_next
ip_in.addr
mux0
+1
+offset
mux1
mux2
mux3
restart_addr_i[9:0]
restart_addr_r [9:0]
load_store_addr_i [9:0]
load_store_addr_r [9:0]
store_data_i[16:0]
store_data_r [16:0]
SRAM
FIFO
instruction_addr_o[9:0]
instruction_data_o[16:0]
ip_in.instr
deque_i
fifo_empty
fifo_full
load_data_o[16:0]
sign extend
offset [9:0]
ram_data [4:0]
Datapath
Address Calculation (AC)
FIFO Store (FS)

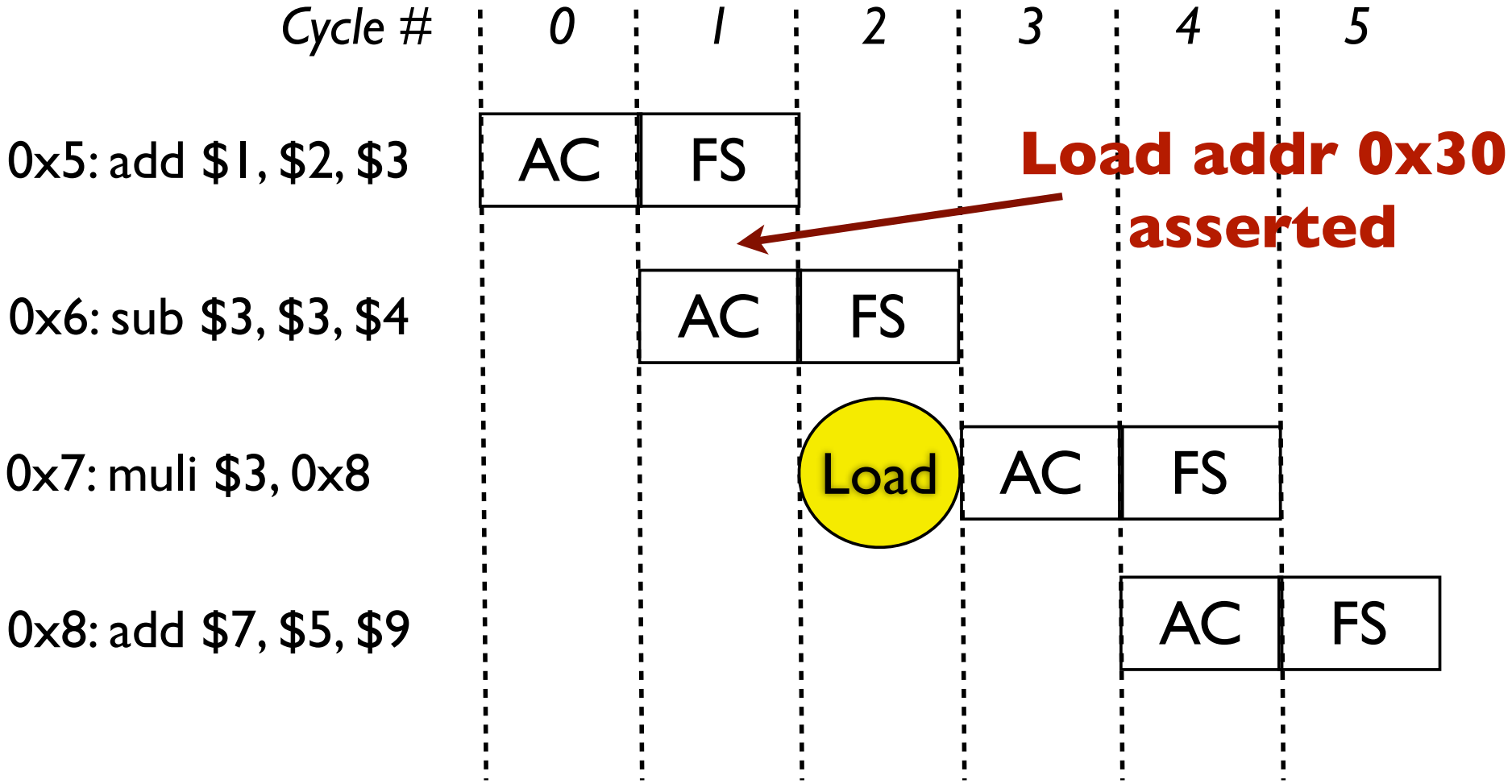
Fetch Pipeline In Action



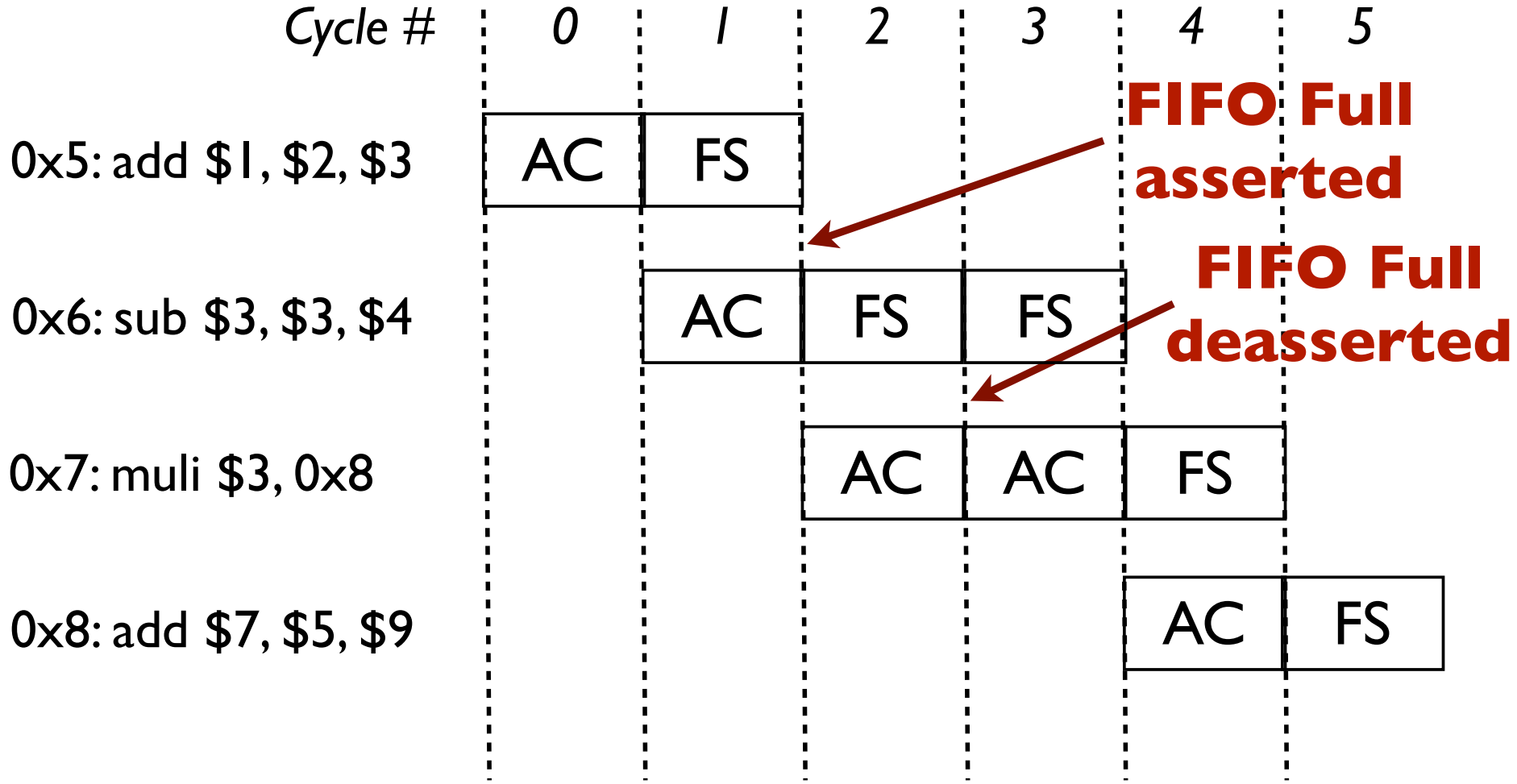
Restarting Fetch



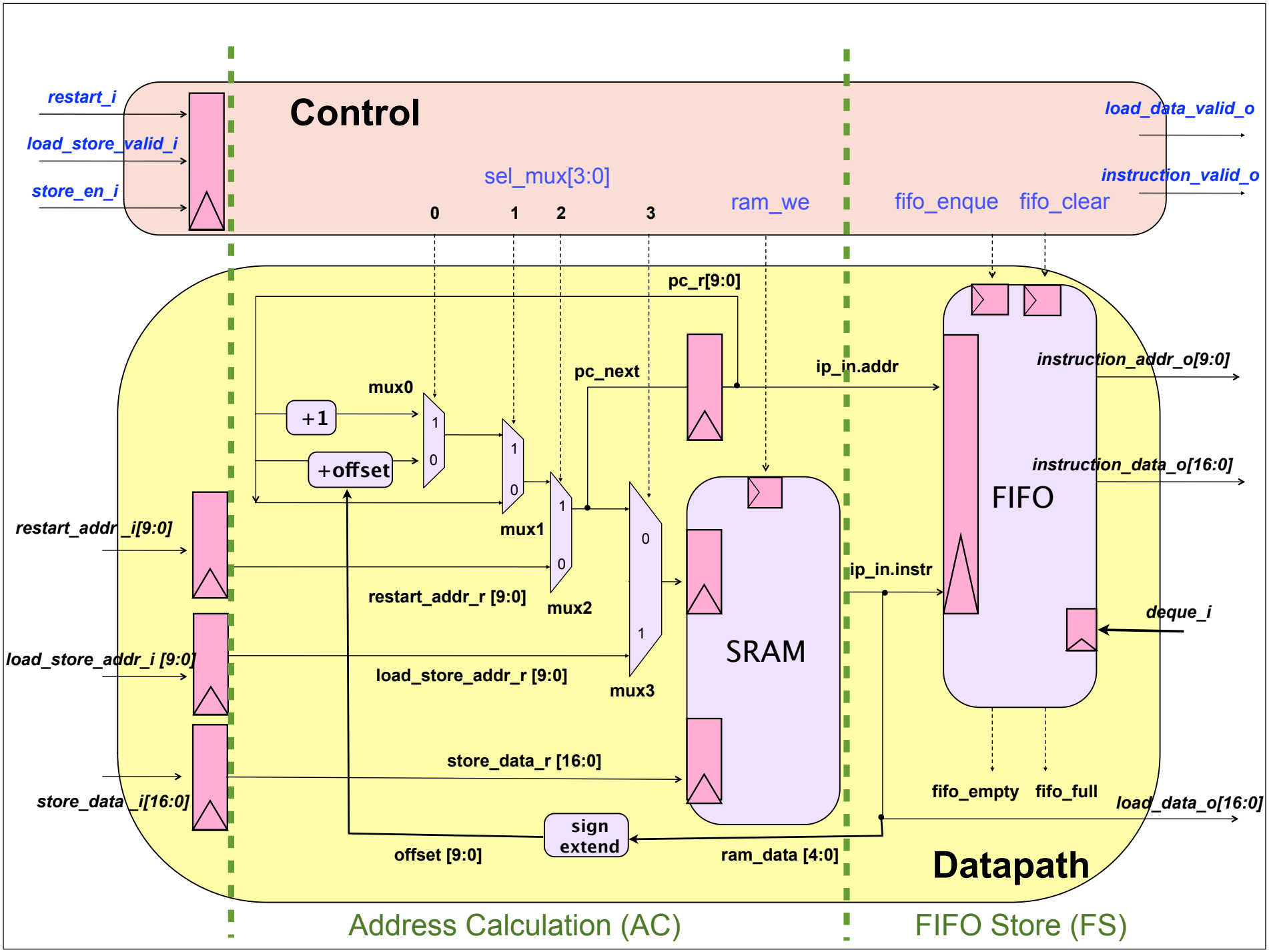
Stalling for loads/stores



Stalled on Full FIFO



Lab 2B:
Fetch Unit Control


Control
restart_i
load_store_valid_i
store_en_i
sel_mux[3:0]
0
1
2
3
ram_we
fifo_enque
fifo_clear
load_data_valid_o
instruction_valid_o
pc_r[9:0]
mux0
+1
+offset
mux1
mux2
mux3
pc_next
ip_in.addr
instruction_addr_o[9:0]
instruction_data_o[16:0]
FIFO
restart_addr_i[9:0]
restart_addr_r [9:0]
load_store_addr_i [9:0]
load_store_addr_r [9:0]
store_data_i[16:0]
store_data_r [16:0]
SRAM
ip_in.instr
deque_i
fifo_empty
fifo_full
load_data_o[16:0]
sign extend
offset [9:0]
ram_data [4:0]
Datapath
Address Calculation (AC)
FIFO Store (FS)

Control logic implementation

- Fetch unit is pipelined
  - Not modeled as a finite state machine
  - Each pipeline stage depends on a subset of the control signals
    - Need to identify which signals those are

Simplified Implementation Stategy

- For each pipeline stage, ask yourself the following questions:
  - What control signals are needed for the datapath during this stage?
  - What does each of these signals do?
  - What inputs do I need to correctly set those control signals?
- Encode relationship between control input and output using Verilog

A Simple Example

- Address Calculation stage:
  - What are the control signals associated with that stage?
    - sel_mux[3:0], ...
  - What does sel_mux[0] do?
    - Selects between PC+1 and PC+offset
  - What is value of sel_mux[0] based on?
    - “P” bit: dictactes whether we branch or not
    - Anything else?