Discussion 8

Pipeline Hazards, MOS Switch, CMOS Logic

Pipeline hazards

- Structural Hazard
  - An instruction in the pipeline may need a resource being used by another instruction in the pipeline
  - Solution: Stalling newer instruction, or adding more hardware
- Data Hazard:
  - An instruction may depend on a data value produced by an earlier instruction
- Control Hazard (branches, exceptions)
  - An instruction may depend on the next instruction’s address

Structural hazard

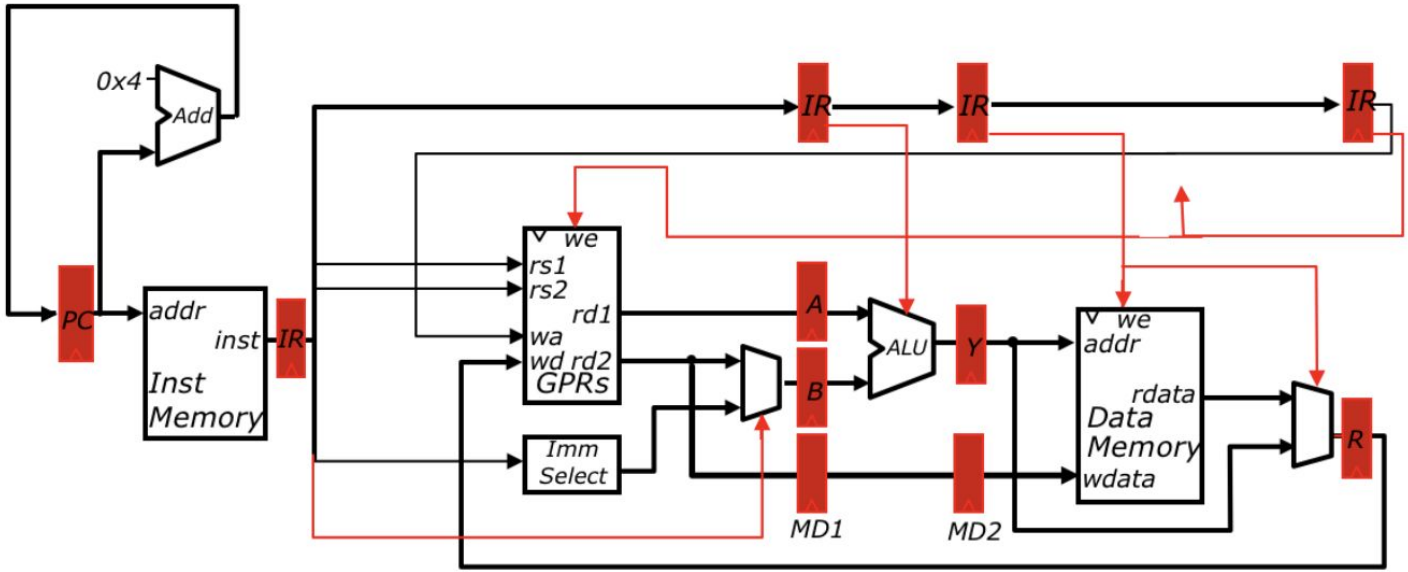
- Memory access
  - Register files have multiple ports (2 read, 1 write)
  - Separate instruction memory and data memory

Data Hazard

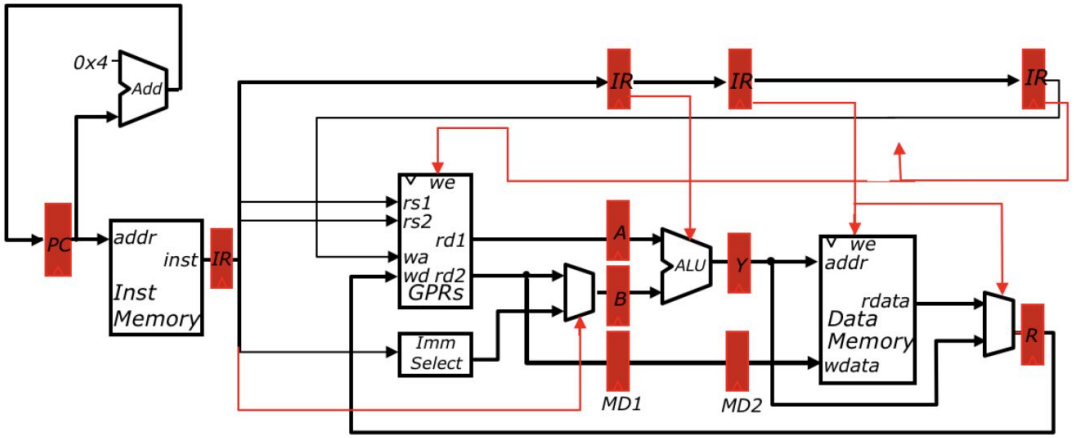
x1 <- x0 + 10

x4 <- x1 + 3

x5 <- x1 + 5

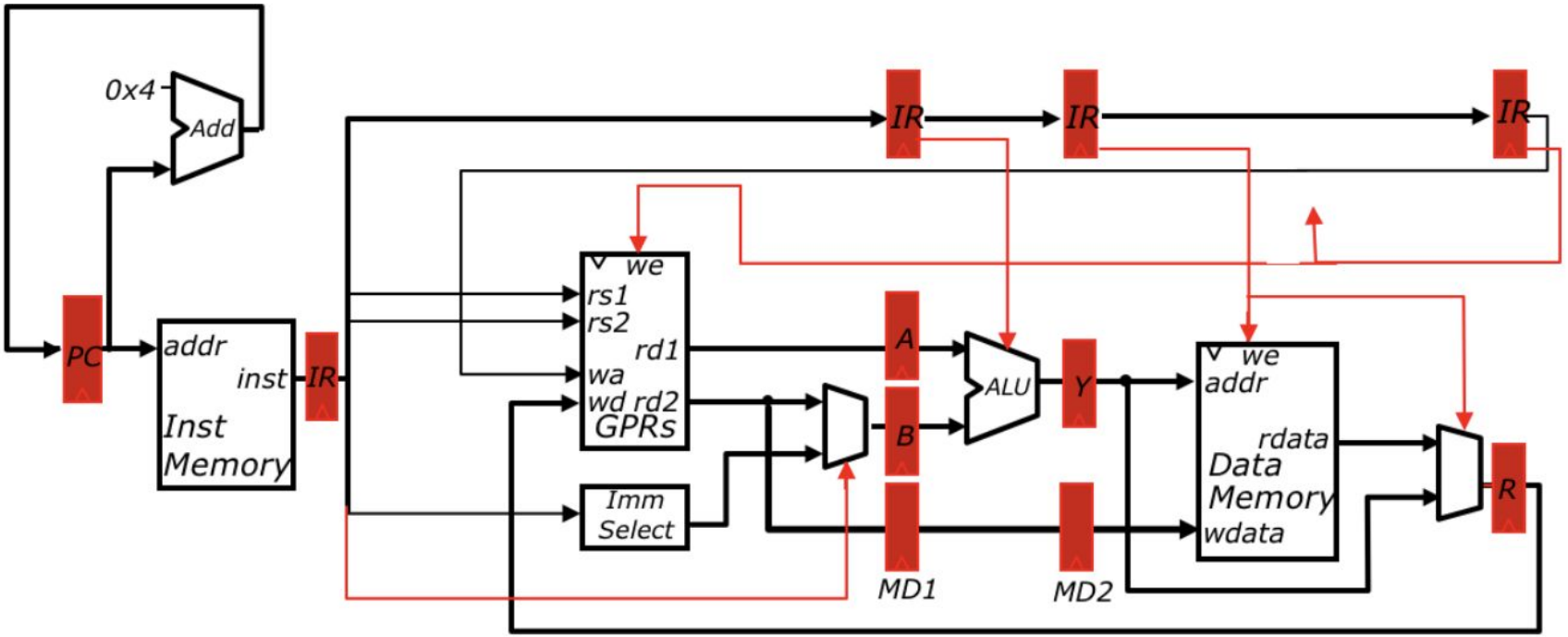


Data Hazard

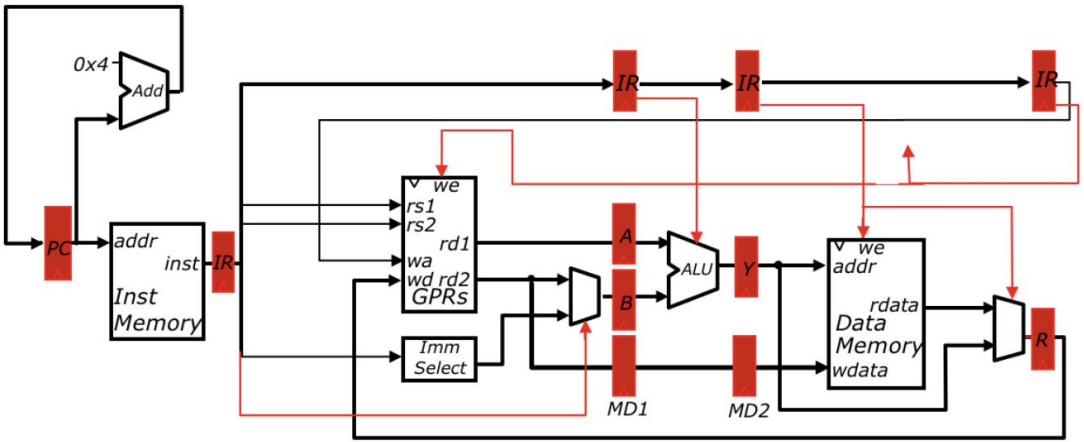


Time	x1 <- x0 + 10	x4 <- x1 + 3	x5 <- x1 + 5
t0	IF		
t1	ID	IF	
t2	EX	ID	IF
t3	MA	?	ID
t4	WB		?
t5			
t6			
t7			

Data Hazard: Stalling



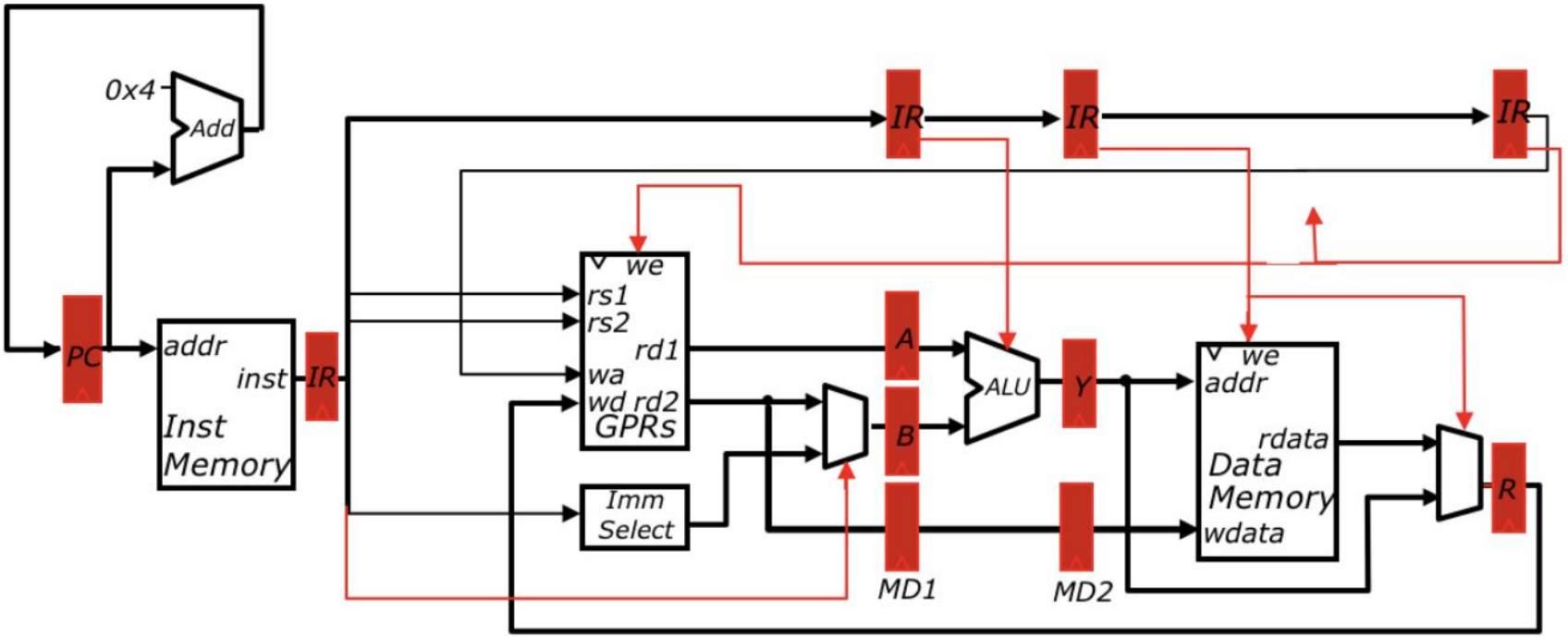
Data hazard: stalling



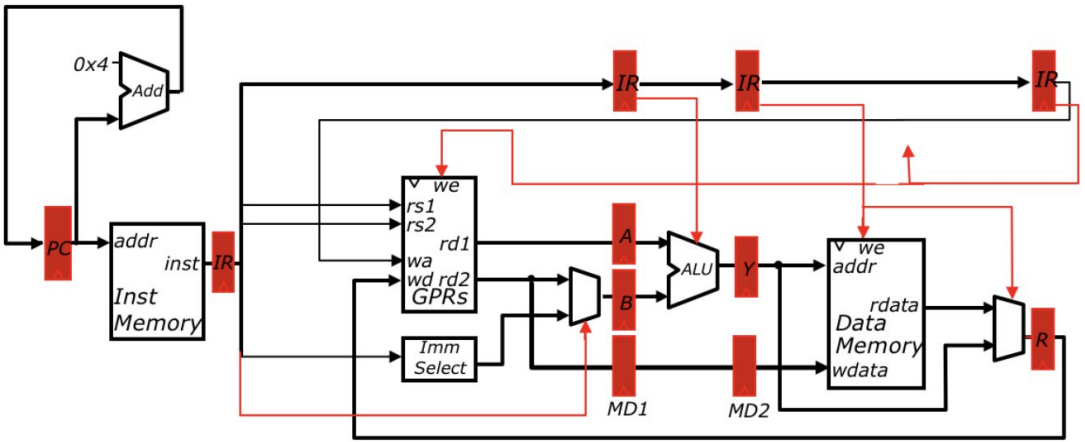
Assume register file can be written and read at the same time

Time	x1 <- x0 + 10	x4 <- x1 + 3	x5 <- x1 + 5
t0	IF		
t1	ID	IF	
t2	EX	ID	IF
t3	MA	ID	IF
t4	WB	ID	IF
t5		ID	IF
t6		EX	ID
t7		MA	EX

Data Hazard: forwarding

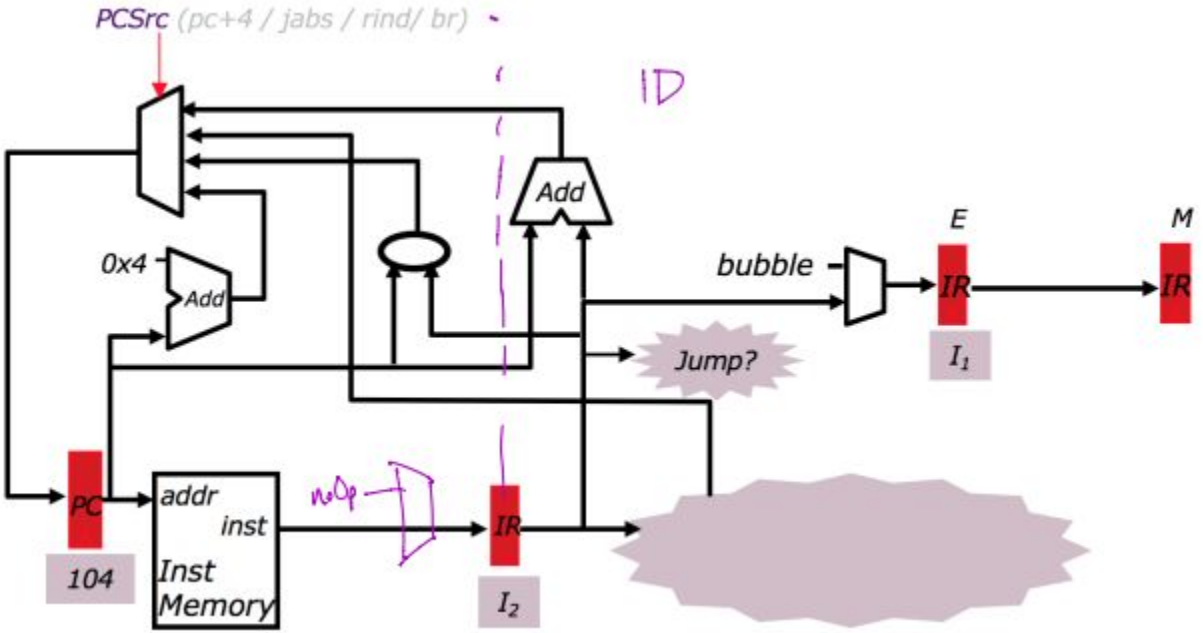


Data Hazard: forwarding



Time	x1 <- x0 + 10	x4 <- x1 + 3	x5 <- x1 + 5
t0	IF		
t1	ID	IF	
t2	EX	ID	IF
t3	MA	EX	ID
t4	WB	MA	EX
t5		WB	MA
t6			WB
t7			

Control Hazard: jump



Example

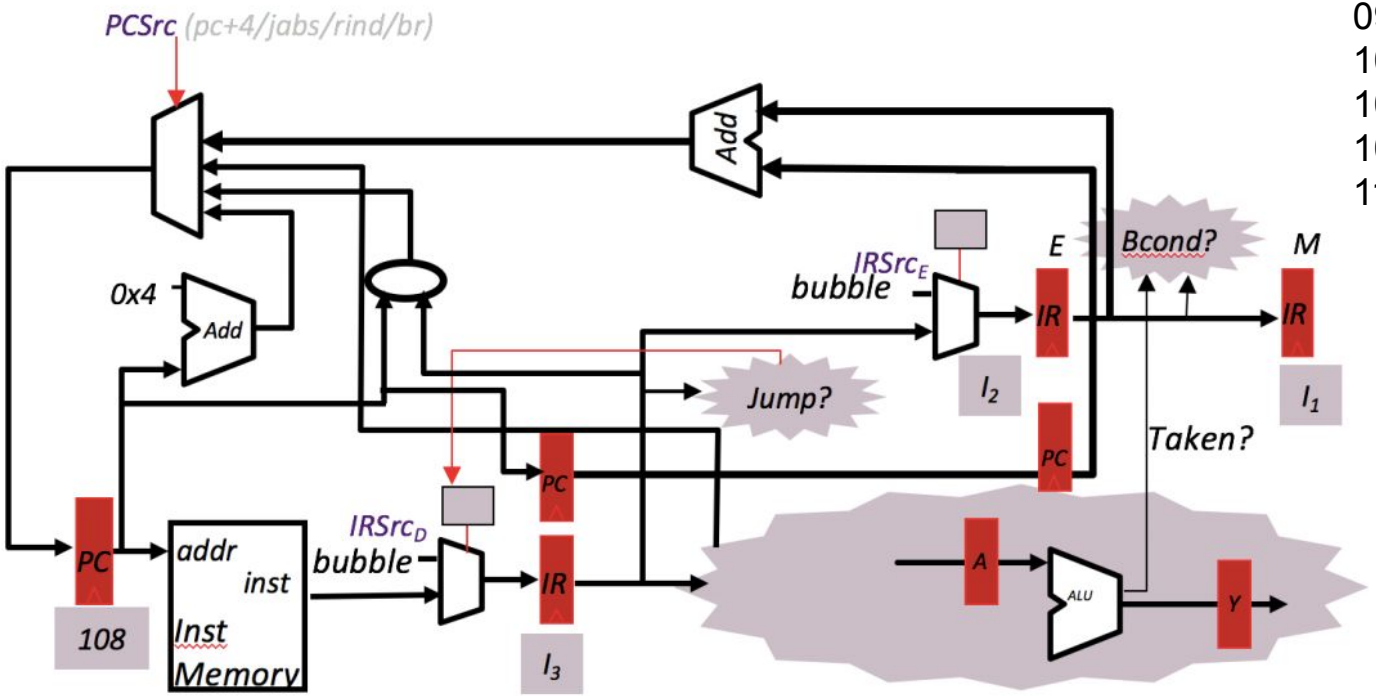
096 ADD
100 J 304
104 ADD
……
304 ADD

Automatically fetched into pipeline, need to kill it

Control Hazard: conditional branch

Example

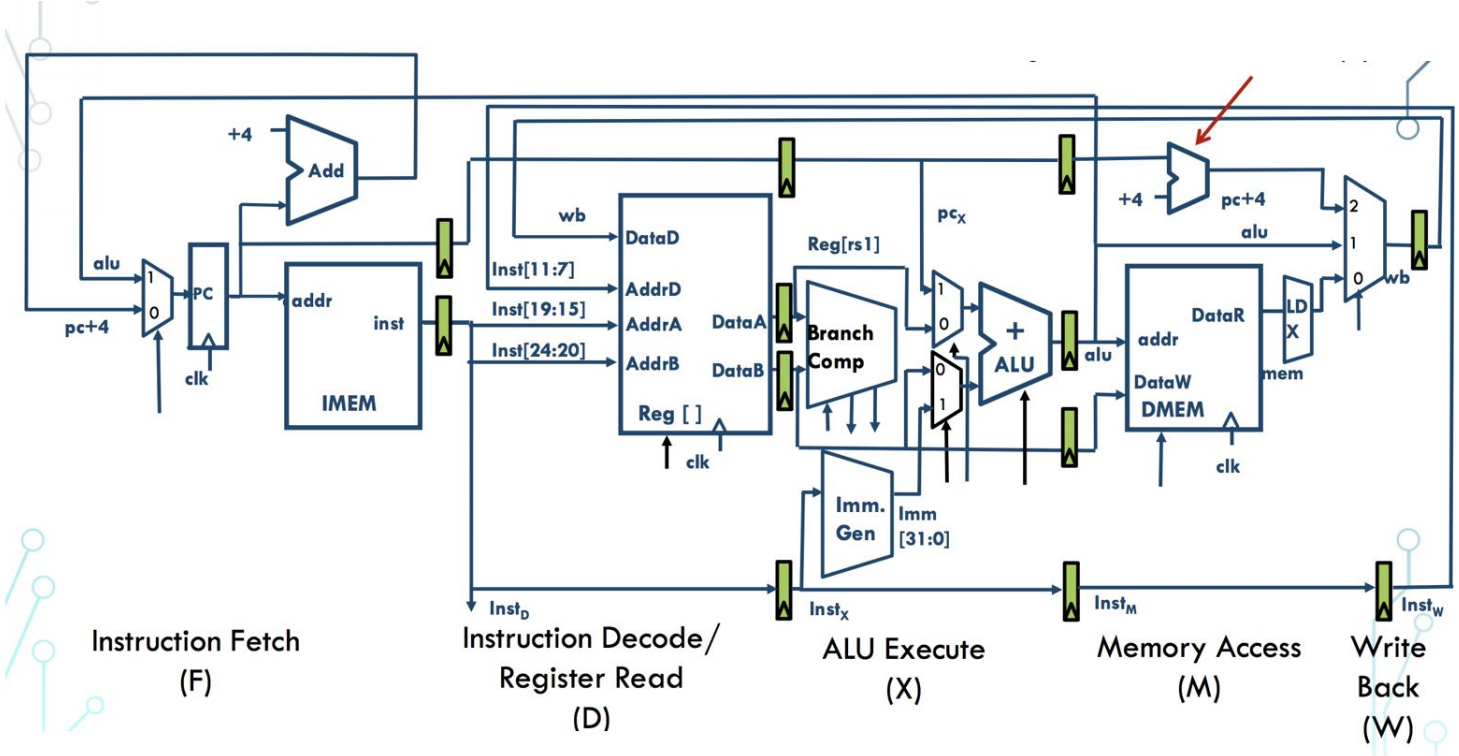
096 ADD
100 BEQ x1, x2, 200
104 ADD
108 ADD
112 ADD



Forwarding vs. Stalling

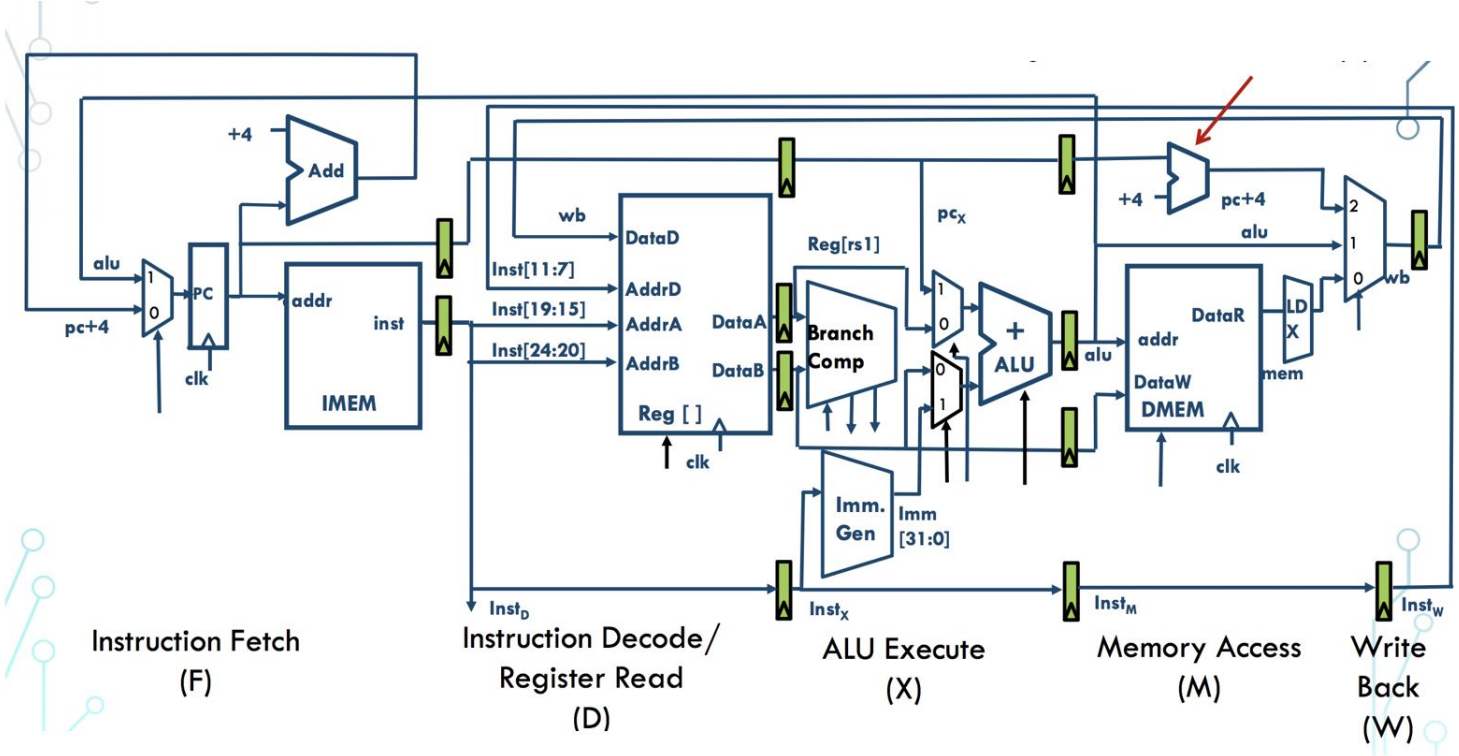
- There are many ways to implement a combination of forwarding or stalling in a pipelined datapath
- You have to find the right tradeoff between a longer critical path, balancing pipeline stage delays, and pipeline stalled cycles
- Consider an Load -> ALU dependency
  - Could forward the load result the same cycle and incur a long critical path but have no stall cycles
  - Could stall the ALU instruction 1 cycle until the load result is in a pipeline register but get a shorter critical path
  - What is better? Try both and see!
  - Intuitively the longer critical path would make the pipeline stages very unbalanced and would hurt max frequency (fmax) more than it would help latency (CPI)
- Iron Law: Execution time = # Insns * CPI * T_clk

HW5 #3 Branch Clarification



- What's the resolution latency of a branch instruction for the 5-stage pipeline from lecture? How many instructions need to be killed?

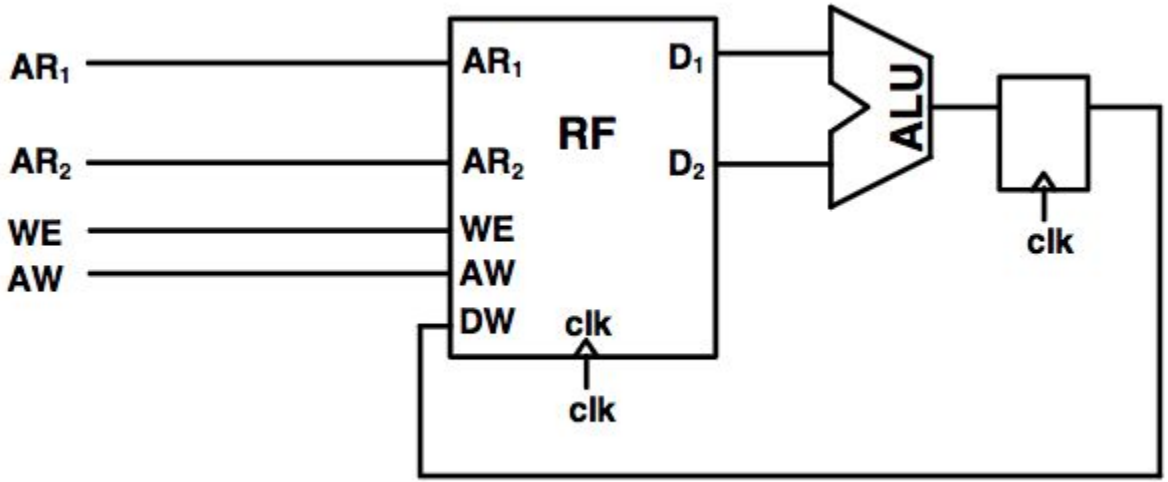
HW5 #3 Branch Clarification



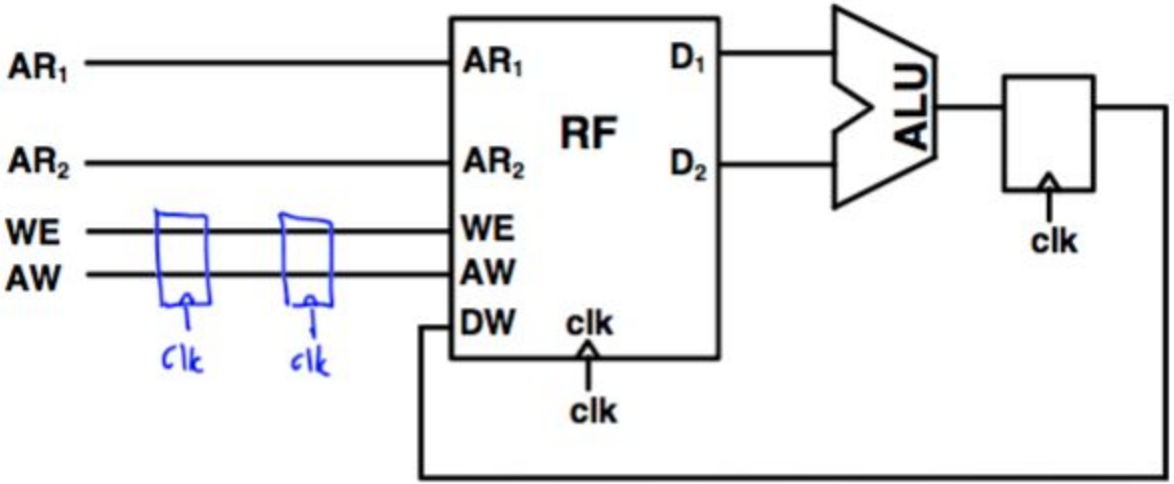
- How should we implement a branch always taken naive predictor? Waiting until the ALU for the branch target address doesn't make sense.

Example

Shown below is a portion of a prototype design for a pipelined CPU's datapath that uses a register file with **synchronous reads and writes**. However, even ignoring any potential data hazards, this design does not function correctly - in particular, register type instructions. Explain what the error is caused by and add any extra components necessary to correct the design.

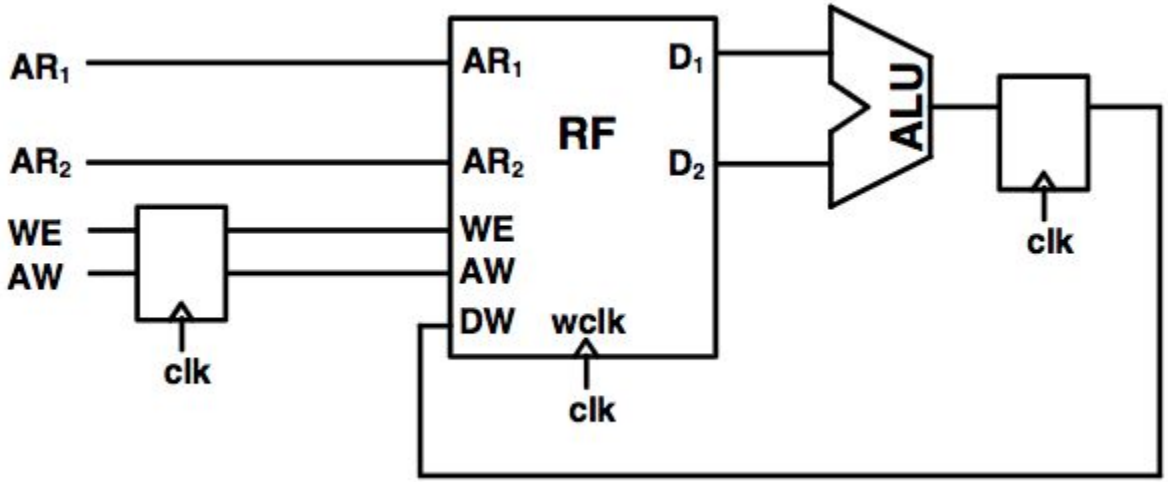


Example

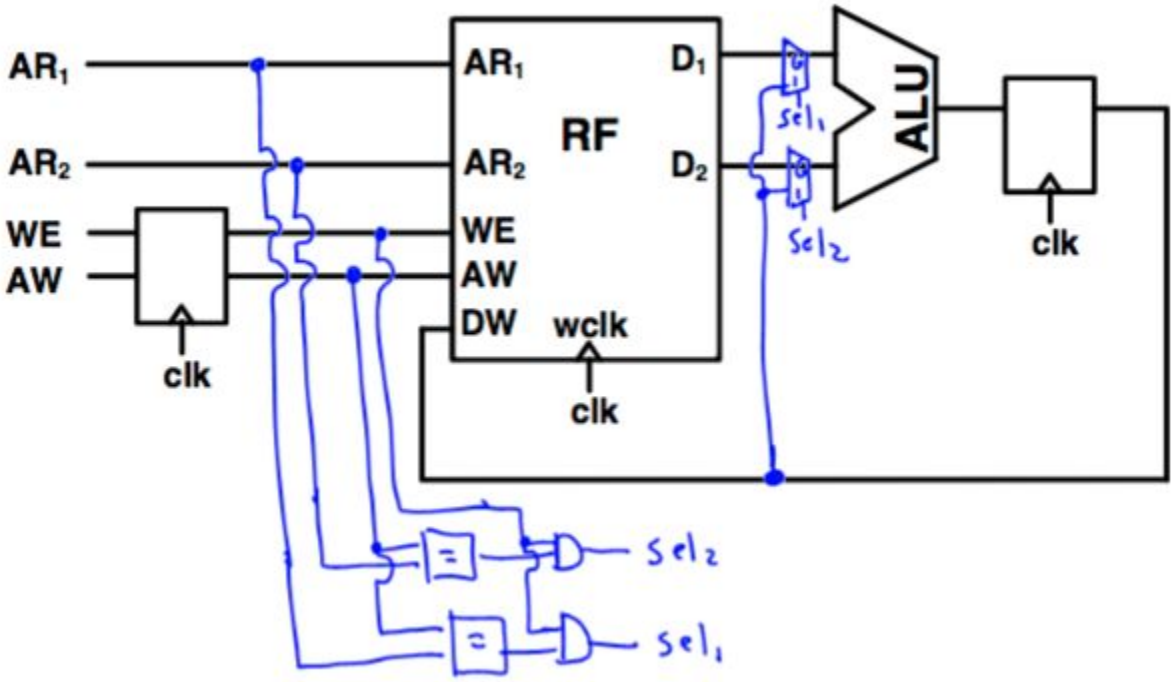


Example

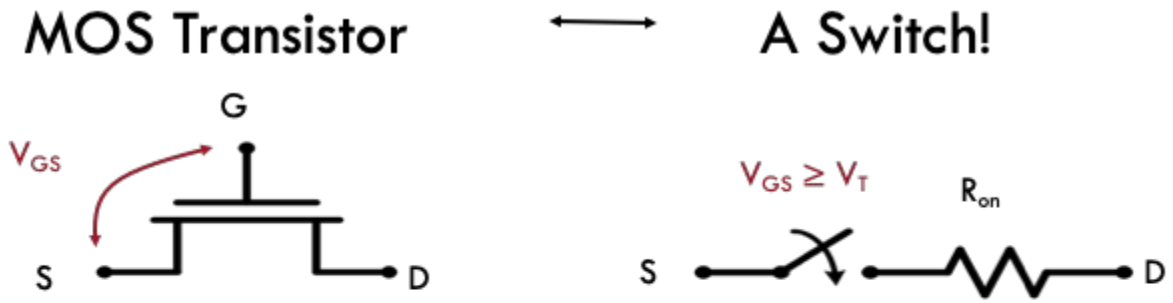
Now this register file has synchronous writes with asynchronous reads. Add appropriate forwarding to eliminate all data hazards.



Example 2


AR1
AR2
WE
AW
clk
AR1
AR2
WE
AW
DW
RF
wclk
clk
D1
D2
sel1
sel2
ALU
clk
sel2
sel1

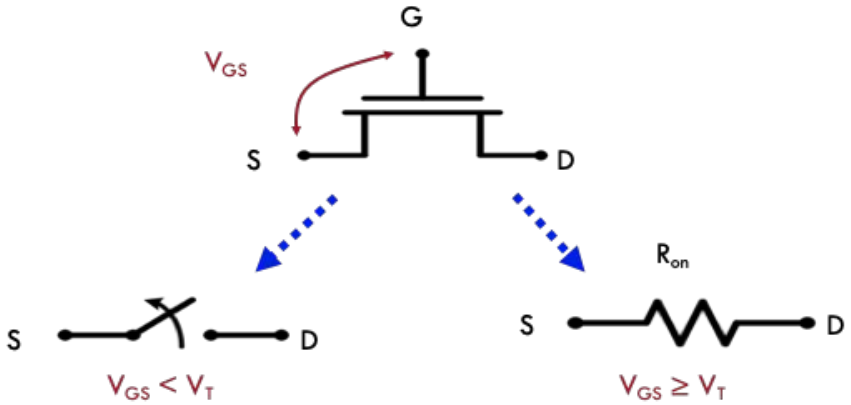
MOS Switch



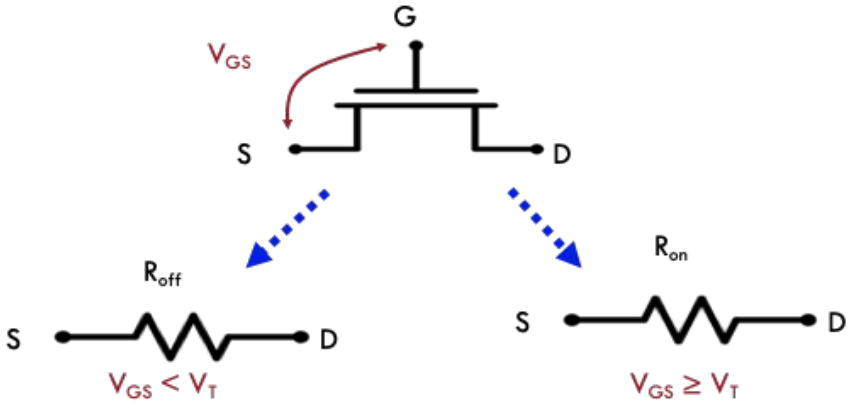
- V_{GS} controls the switch
  - (it also charges the channel capacitor)

MOS Switch

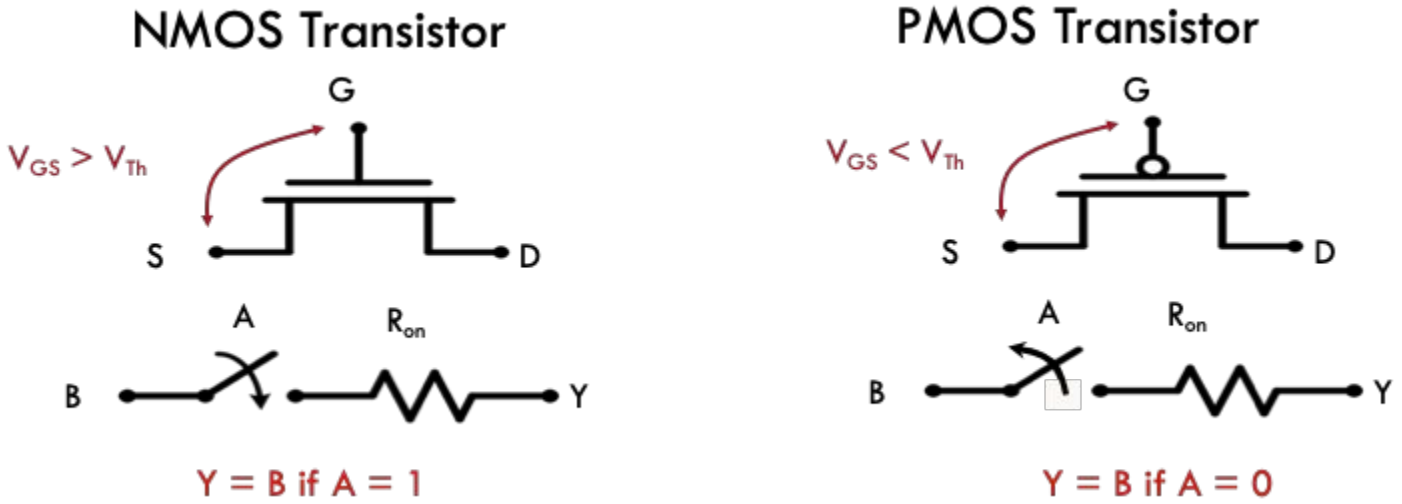
ON/OFF Model



“More Realistic” Model

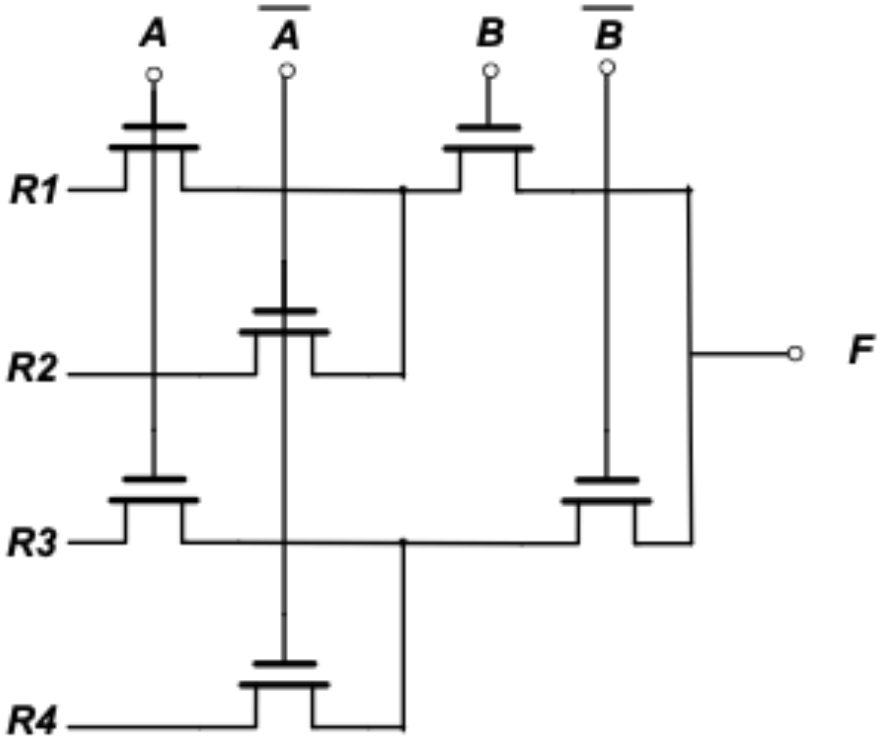


MOS Switch



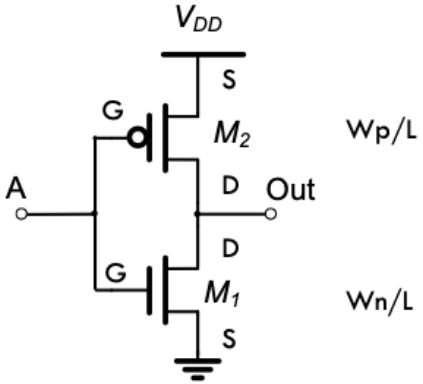
- Source of NMOS always at lower voltage
- Source of PMOS always at higher voltage
- The ‘effective’ source node can change depending on the voltage at the MOS’ terminals

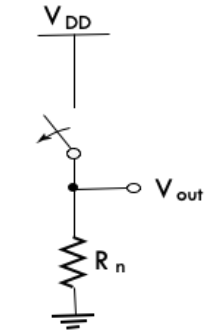
Switch Logic



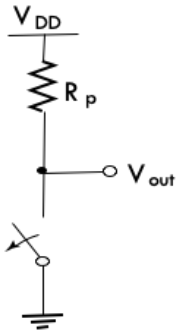
CMOS Inverter

- Schematic



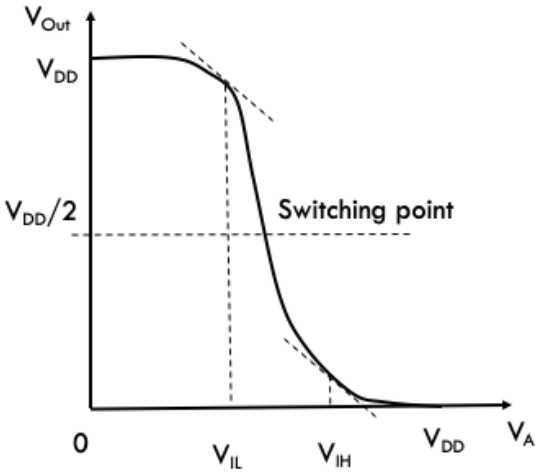


$V_{in} = V_{DD}$



$V_{in} = 0$

$V_{OL} = 0$
$V_{OH} = V_{DD}$

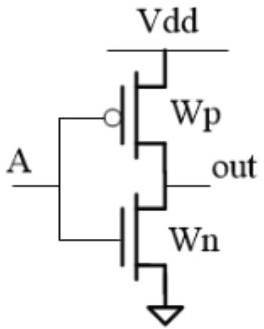


Inverter VTC

[7pts/11pts] Voltage Transfer Characteristics (VTCs).

The technology has the following parameters: $V_{th,N} = 0.2V$ and $|V_{th,P}| = 0.3V$, $R_n = 2k\Omega * \mu m$, $R_p = 3k\Omega * \mu m$ at $V_{dd} = 1V$. Draw the voltage transfer characteristic (V_{out} vs V_A) of the gates below with $W_p = W_n = 1\mu m$.

(a) [4pts] Draw the VTC and determine V_{OL}, V_{IL}, V_{OH}, V_{IH} and noise margins NM_H and NM_L.



Remember:

Noise margin high:
$NM_H = V_{OH} - V_{IH}$

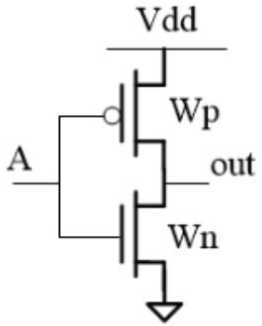
Noise margin low:
$NM_L = V_{IL} - V_{OL}$

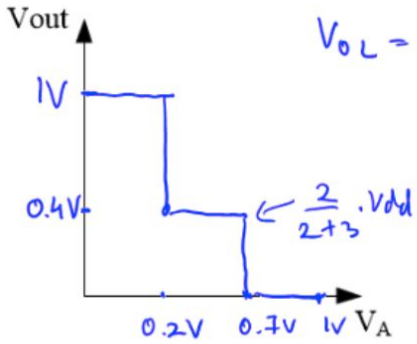
Inverter VTC

[7pts/11pts] Voltage Transfer Characteristics (VTCs).

The technology has the following parameters: $V_{th,N} = 0.2V$ and $|V_{th,P}| = 0.3V$, $R_n = 2k\Omega * \mu m$, $R_p = 3k\Omega * \mu m$ at $V_{dd} = 1V$. Draw the voltage transfer characteristic (V_{out} vs V_A) of the gates below with $W_p = W_n = 1\mu m$.

(a) [4pts] Draw the VTC and determine V_{OL}, V_{IL}, V_{OH}, V_{IH} and noise margins NM_H and NM_L.





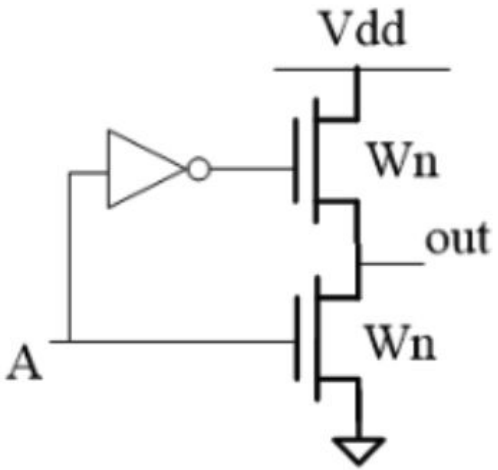
$V_{OH} = 1V$ $V_{IH} = 0.7V$

$V_{OL} = 0V$ $V_{IL} = 0.2V$

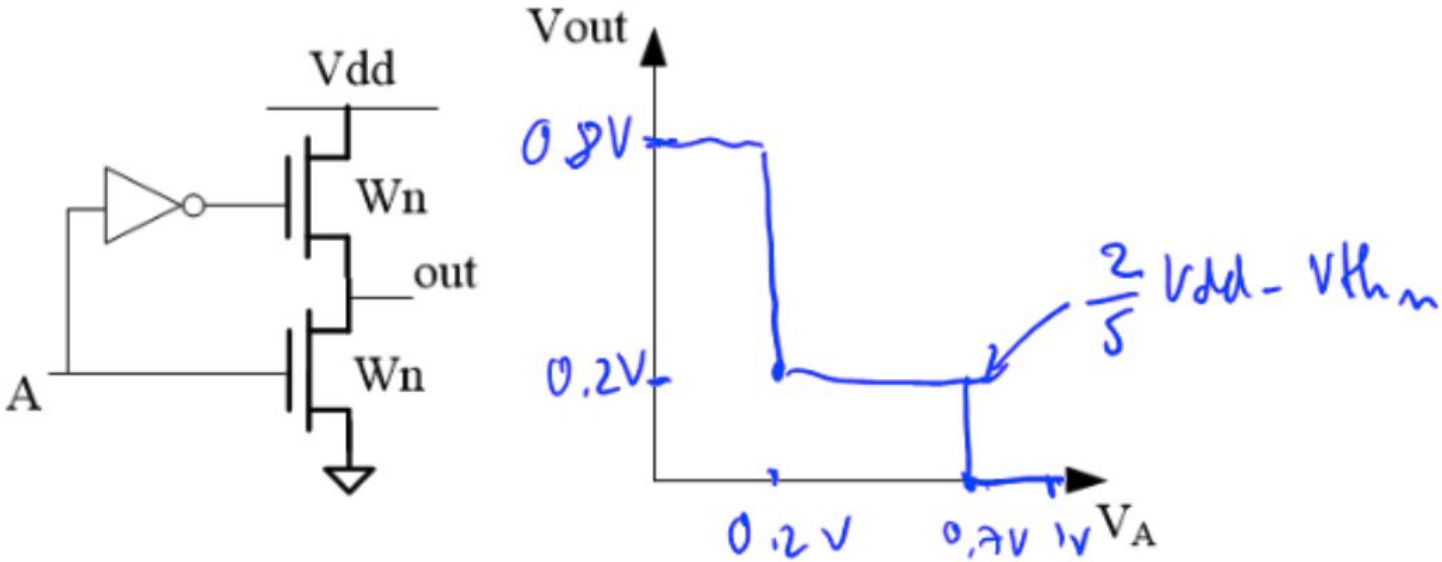
$NM_H = 0.3V$

$NM_L = 0.2V$

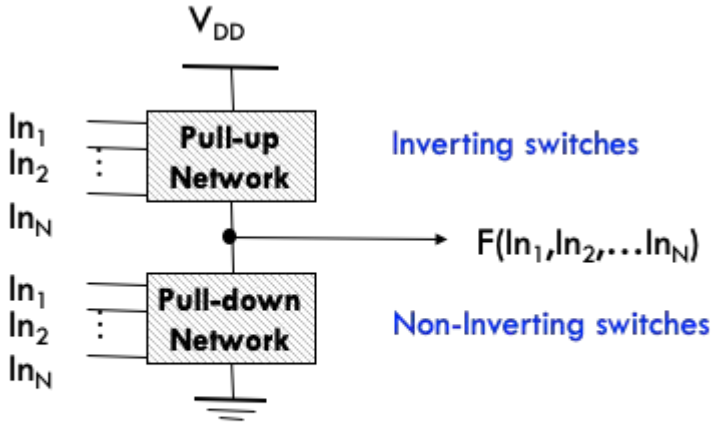
Inverter VTC



Inverter VTC

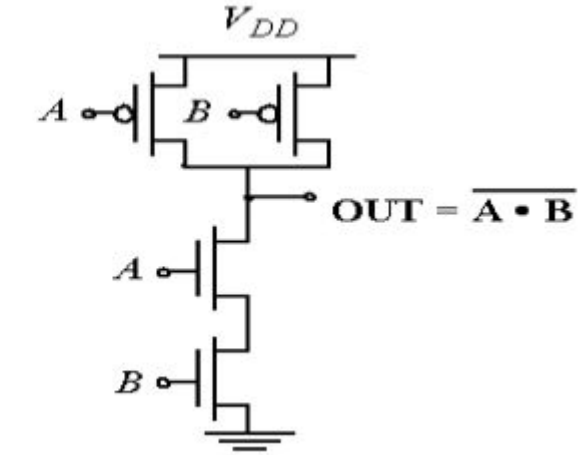


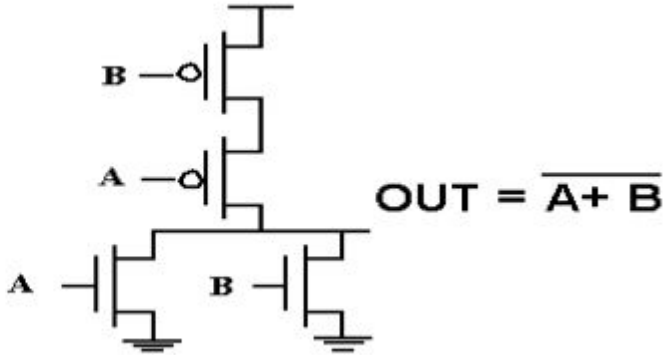
CMOS Logic



PUN and PDN are dual logic networks

PUN and PDN functions are complementary





CMOS Logic

XNOR:

CMOS Logic

AND2: