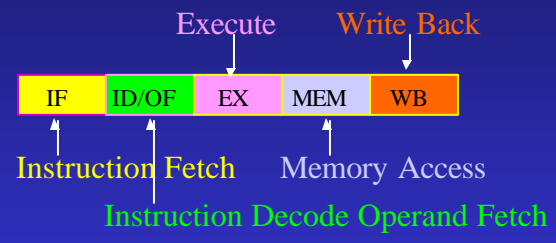
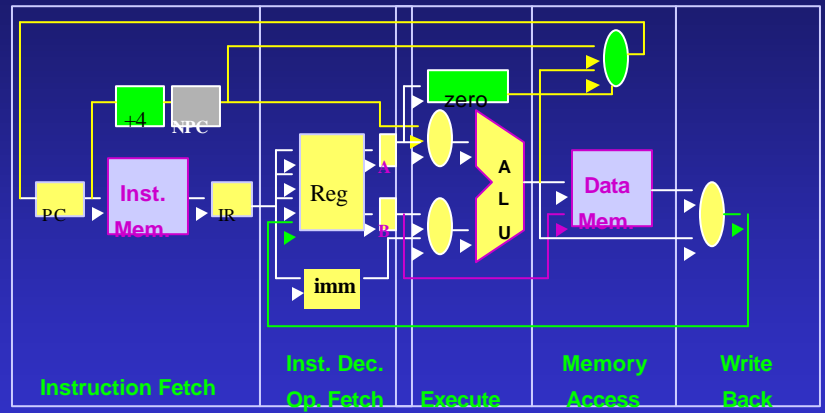


# Processor

## Basic steps to process an instruction

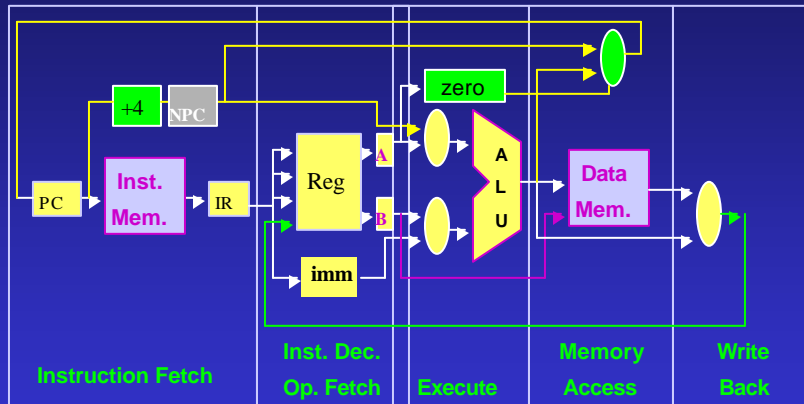


# Datapath



$$\begin{aligned}
 IR &\leftarrow \text{Mem}[\text{PC}] & A &\leftarrow \text{Reg}[\text{IR}_{6..10}] \\
 \text{NPC} &\leftarrow \text{PC} + 4 & B &\leftarrow \text{Reg}[\text{IR}_{11..15}] \\
 \text{Imm} &\leftarrow ((\text{IR}_{16})^{\text{imm}} \# \text{IR}_{11..15})
 \end{aligned}$$

## Datapath (arith/logic inst)

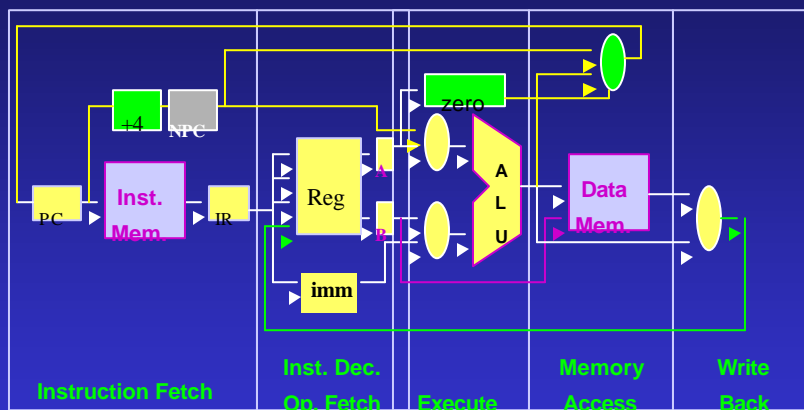


$IR \leftarrow Mem[PC]$   
 $NPC \leftarrow PC + 4$   
 $A \leftarrow Reg[IR_{6..10}]$   
 $B \leftarrow Reg[IR_{11..15}]$   
 $Imm \leftarrow ((IR_{16})^1 \# \# IR_{11..15}]$

EE524/CptS561 Jose G. Delgado-Frias

3

## Datapath (load Inst)

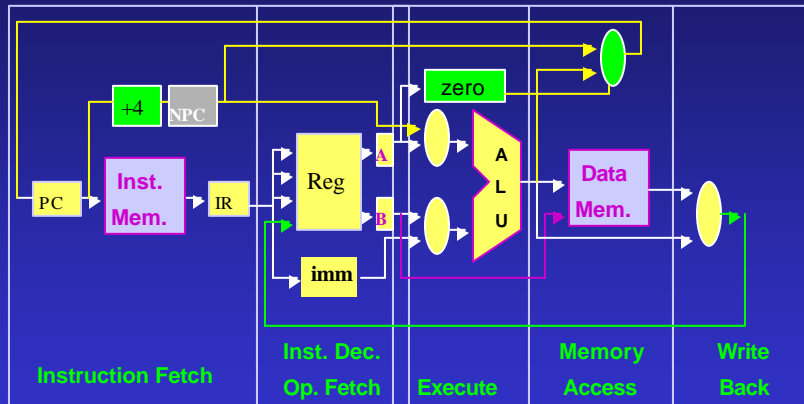


$IR \leftarrow Mem[PC]$   
 $NPC \leftarrow PC + 4$   
 $A \leftarrow Reg[IR_{6..10}]$   
 $B \leftarrow Reg[IR_{11..15}]$   
 $Imm \leftarrow ((IR_{16})^1 \# \# IR_{11..15}]$

EE524/CptS561 Jose G. Delgado-Frias

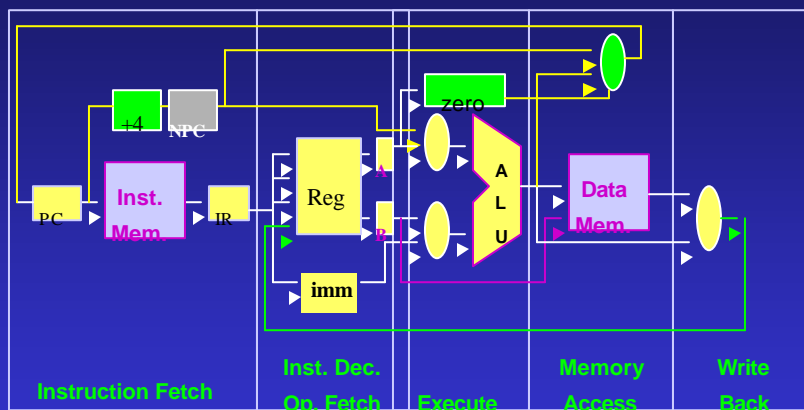
4

## Datapath (store Inst)



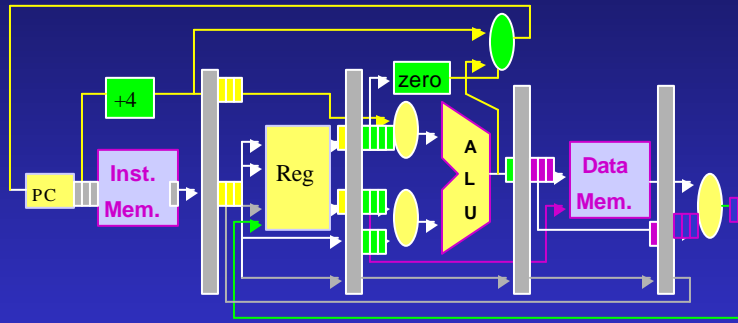
$IR \leftarrow Mem[PC]$   
 $NPC \leftarrow PC + 4$   
 $A \leftarrow Reg[IR_{6..10}]$   
 $B \leftarrow Reg[IR_{11..15}]$   
 $Imm \leftarrow ((IR_{16})^1 \# \# IR_{11..15})$

## Datapath (branch Inst)



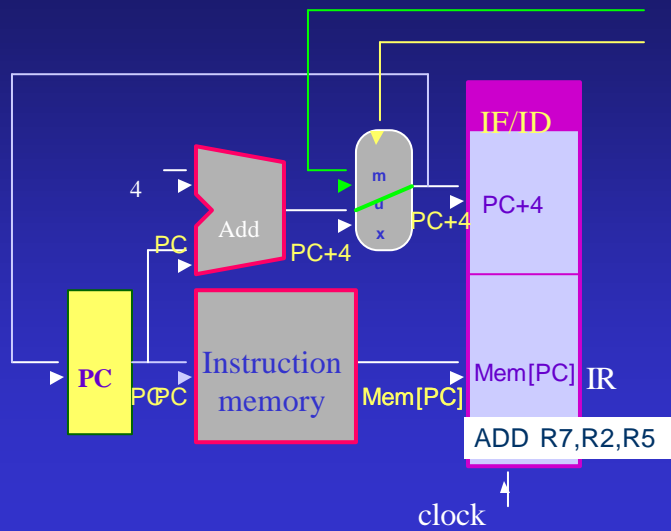
$IR \leftarrow Mem[PC]$   
 $NPC \leftarrow PC + 4$   
 $A \leftarrow Reg[IR_{6..10}]$   
 $B \leftarrow Reg[IR_{11..15}]$   
 $Imm \leftarrow ((IR_{16})^1 \# \# IR_{11..15})$

# Datapath w/ pipeline

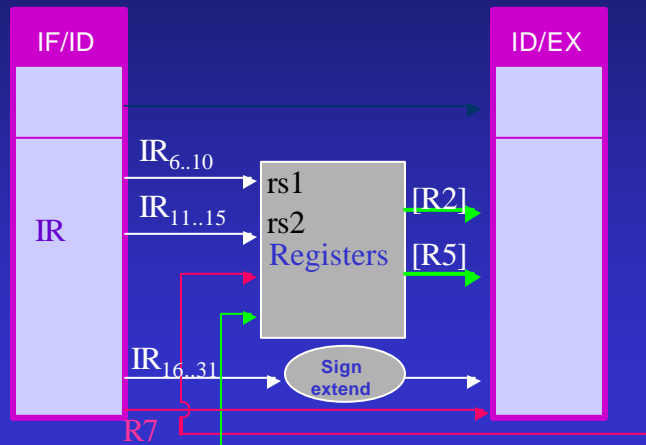


0

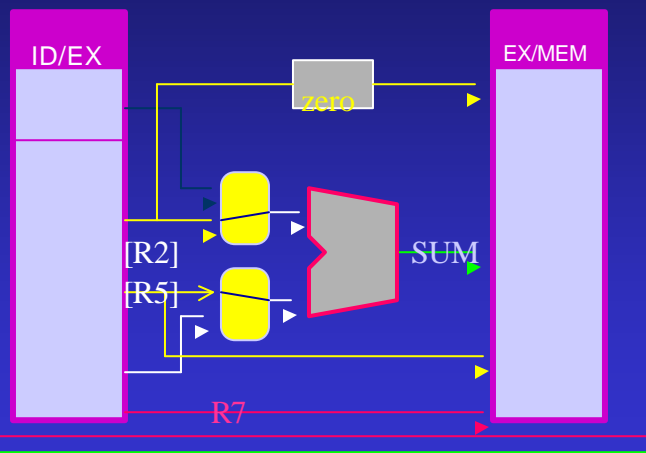
# Instruction Fetch (IF) stage



## Instruction Decode (ID) stage

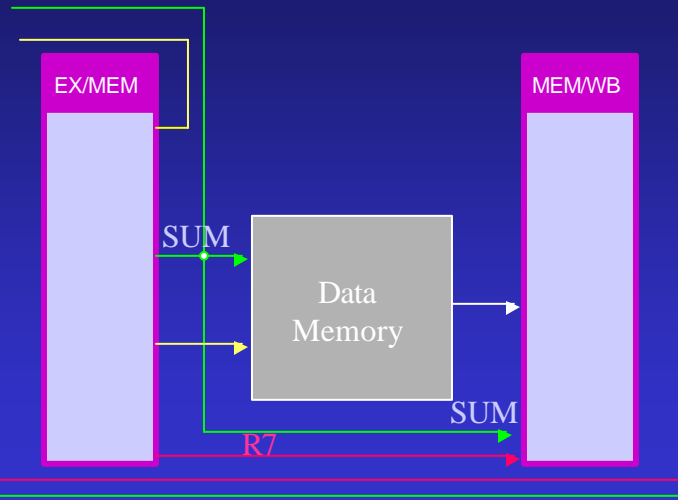


## Execution (EX) stage



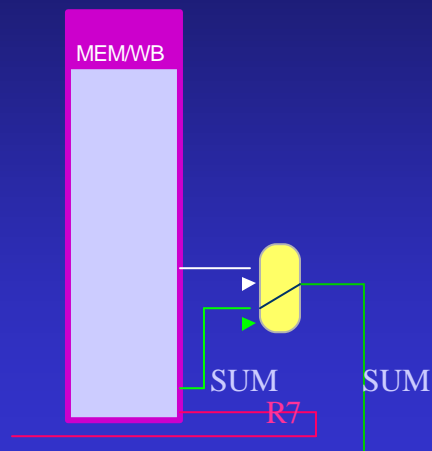
3

# Memory (MEM) stage



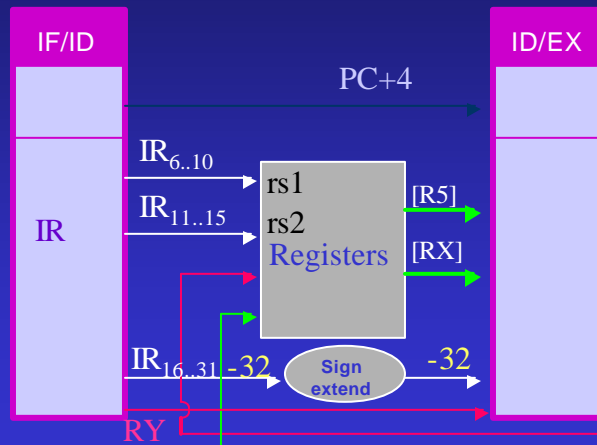
4

# Write Back (WB) stage

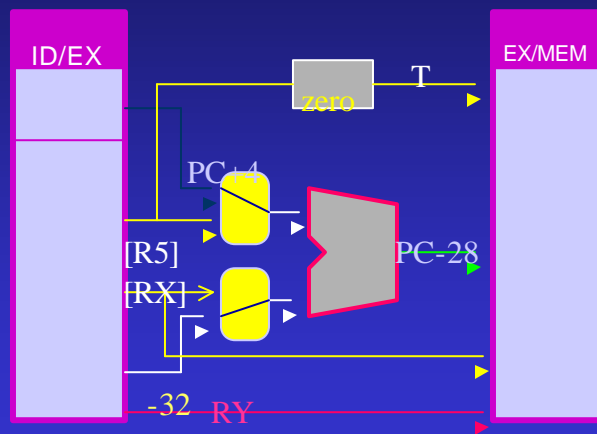




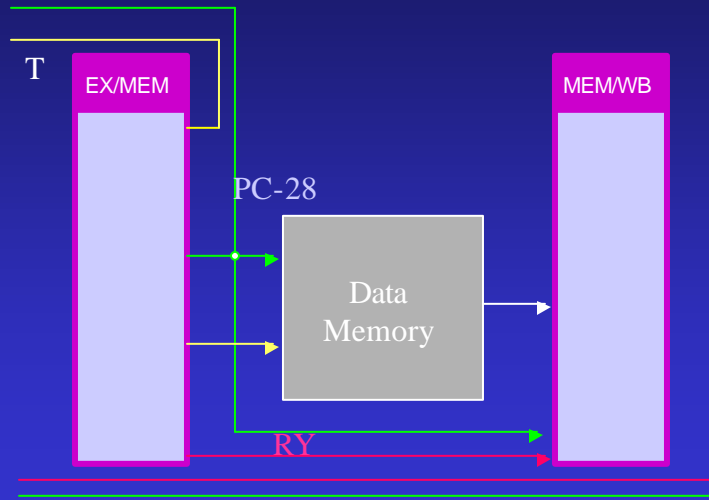
## Instruction Decode (ID) stage



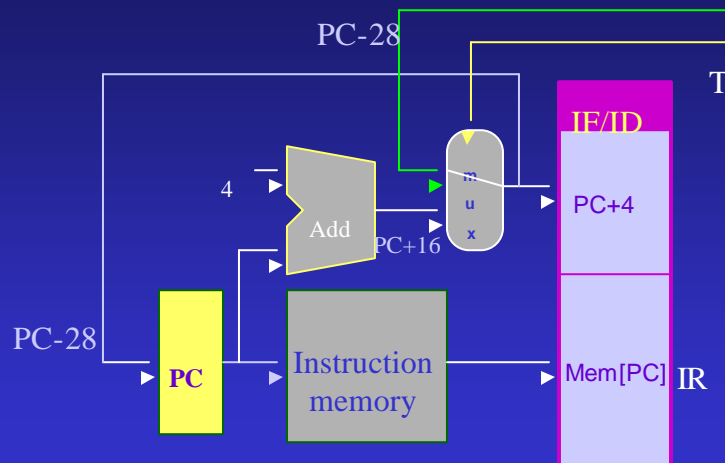
## Execution (EX) stage



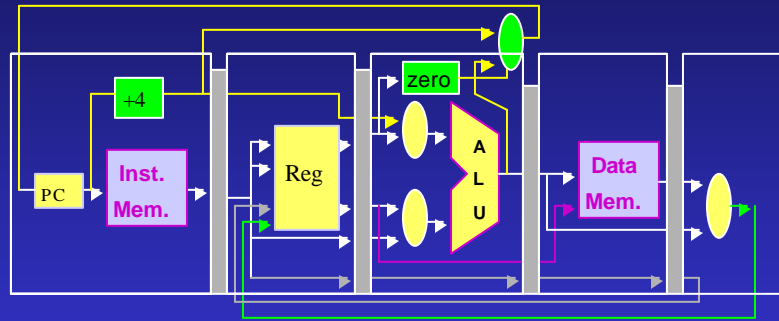
## Memory (MEM) stage



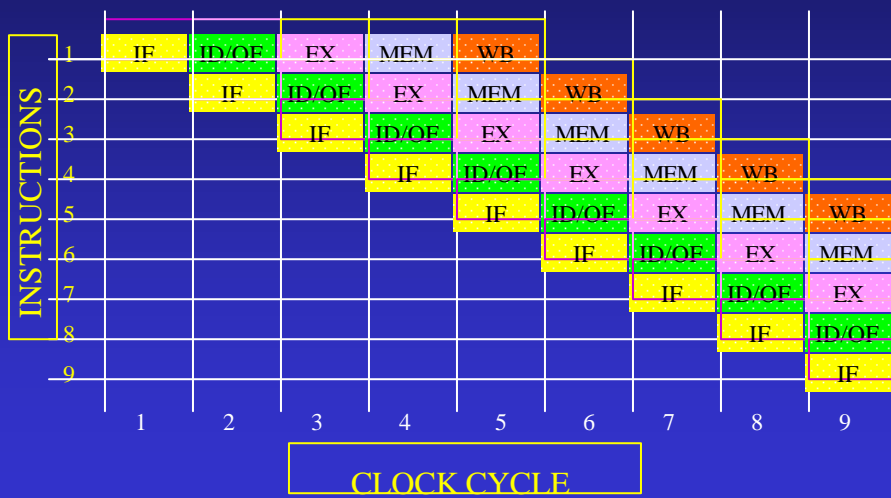
## Instruction Fetch (IF) stage



# Datapath w/ pipeline



# Pipeline



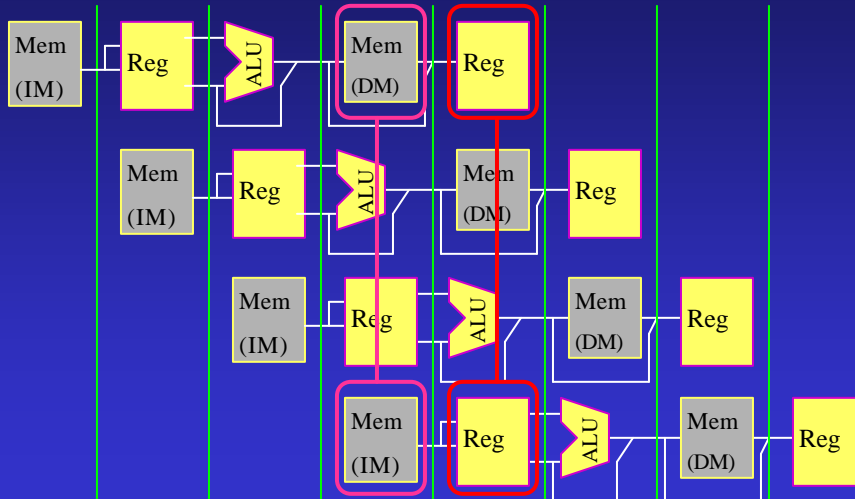
# Pipeline

Clock cycle	1	2	3	4	5	6	7	8
	IF	ID/OF	EX	MEM	WB			
		IF	ID/OF	EX	MEM	WB		
			IF	ID/OF	EX	MEM	WB	
				IF	ID/OF	EX	MEM	WB
					IF	ID/OF	EX	MEM
						IF	ID/OF	EX
							IF	ID/OF
								IF

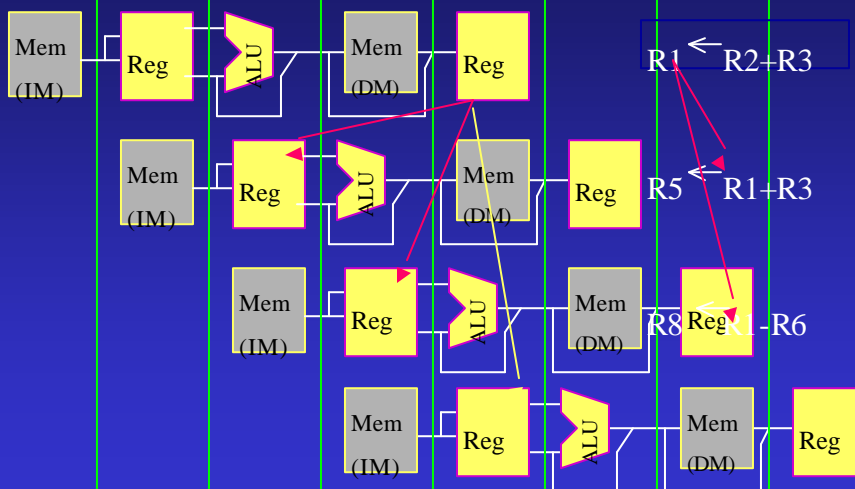
# Pipeline Hazards

- Structural Hazards
  - two or more instructions use same hardware at the same time.
- Data Hazards
  - Data dependencies
  - Result from inst. j is needed by inst. k
- Control Hazards
  - Branch changes flow, what happen with the following instruction(s)

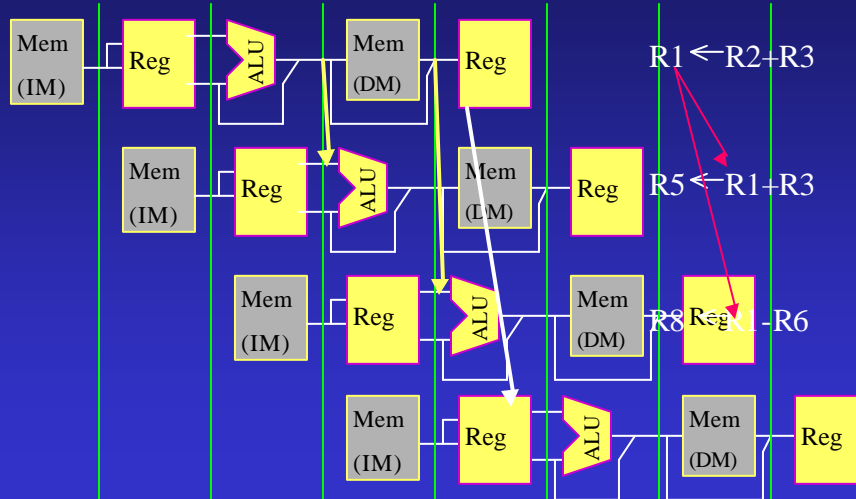
# Resources



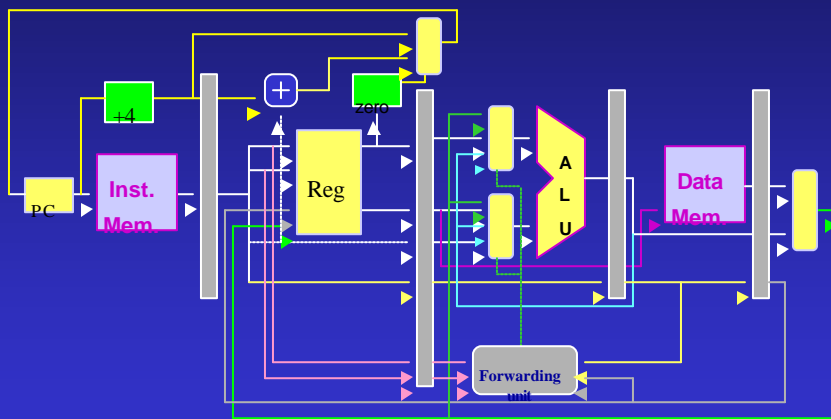
# Data Hazards



# Data Forwarding

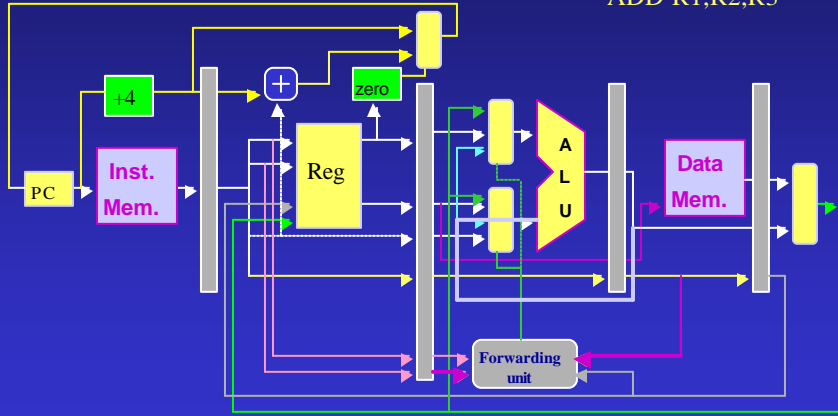


# Datapath w/ pipeline



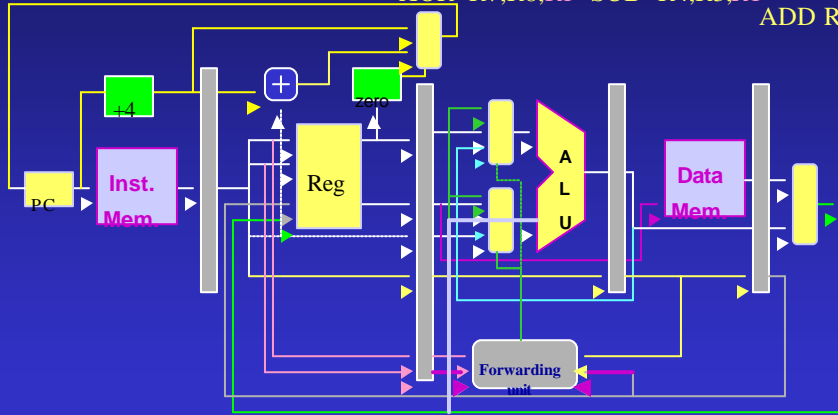
# Example

XOR R7,R8,R1    XOR R4,R3,R3    SUB R4,R3,R3    ADD R1,R2,R3



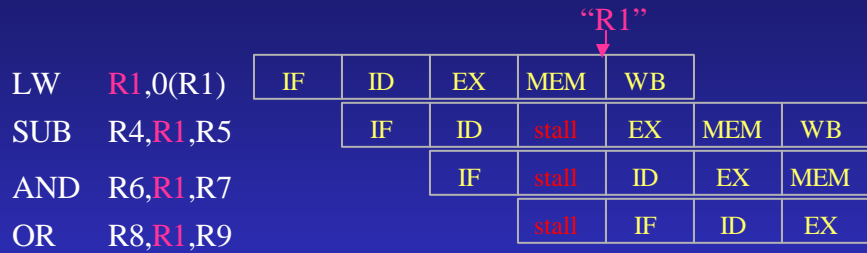
# Example

XOR R7,R8,R1    SUB R4,R3,R1    ADD R1..





## Data hazard (load)

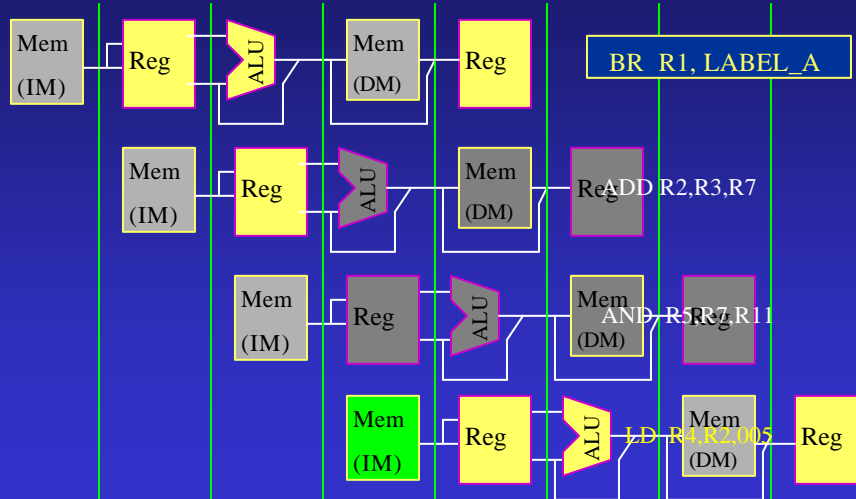


## Branch

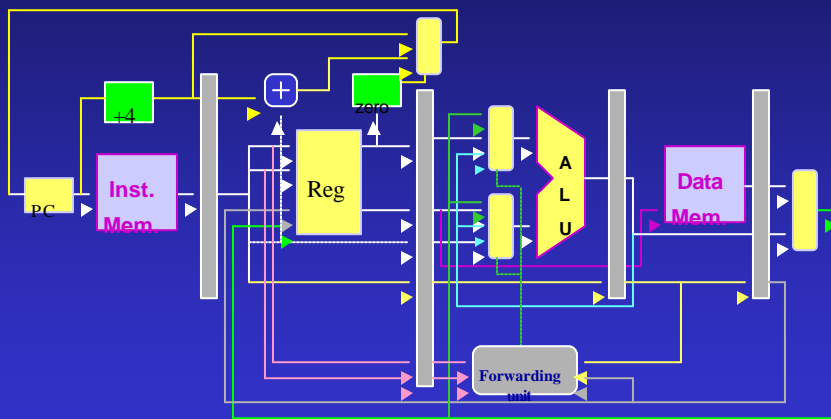
```

BR R1, LABEL_A
ADD R2,R3,R7
AND R5,R7,R11
:
:
LABEL_A: LD R4,R2,005
    
```

# Branch



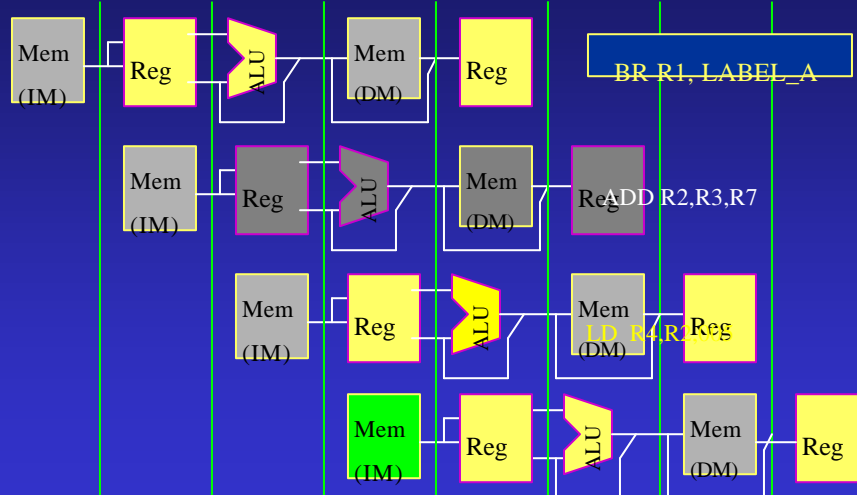
# Datapath w/ pipeline



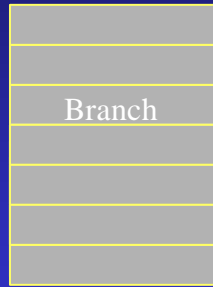
# What to do w/ branch

- Reduce the number of cycles to decide on a branch.
- **Delayed branch (Software Solutions)**
  - NO-OP
  - **move instructions**
    - from before
    - from target
    - from fall through

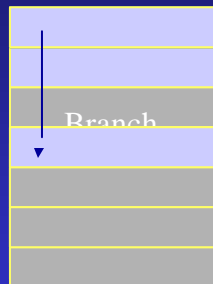
## Branch



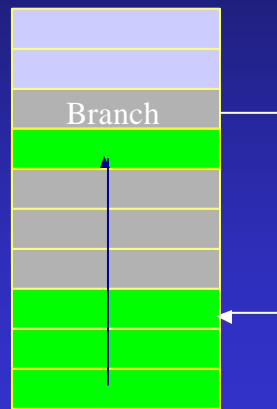
# NO-OP



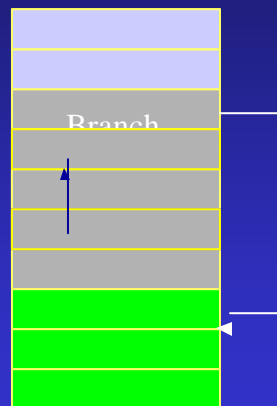
# From Before



# From Target



# From Fall Through



## Example

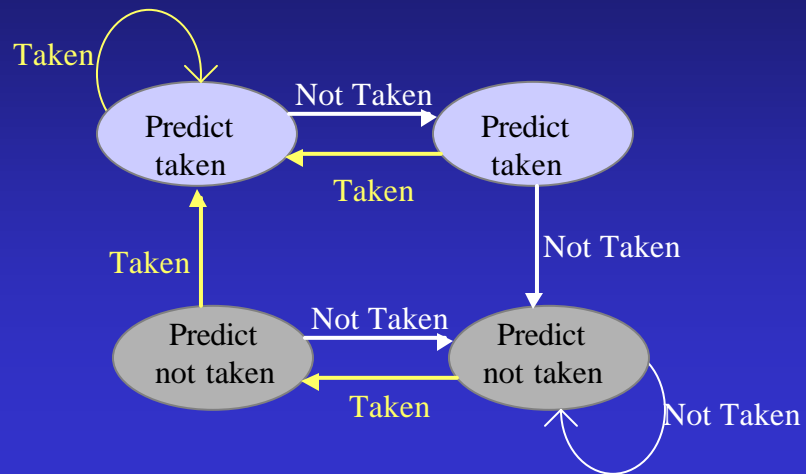
<b>loop:</b>	<b>LW</b>	<b>R1,0(R2)</b>	<b>R1 ← MEM[R2+0]</b>
	<b>ADDI</b>	<b>R1,R1,#1</b>	<b>R1 ← R1+1</b>
	<b>SW</b>	<b>0(R2),R1</b>	<b>MEM[R2+0] ← R1</b>
	<b>ADDI</b>	<b>R2,R2,#4</b>	<b>R2 ← R2+4</b>
	<b>SUB</b>	<b>R4,R3,R2</b>	<b>R4 ← R3-R2</b>
	<b>BNEZ</b>	<b>R4,loop</b>	<b>IF R4 ≠ 0 GOTO loop</b>

Initial value:  $R3 = R2 + 396$

## Dynamic Branch Prediction

- Hardware approach
- Based on past history
  - 2-bit counter

## 2-bit prediction scheme



EE524/CptS561 Jose G. Delgado-Frias

43

## Branch Target Buffer (BTB)

Current PC	Predicted PC	

Branch Prediction  
Taken / Untaken

EE524/CptS561 Jose G. Delgado-Frias

44