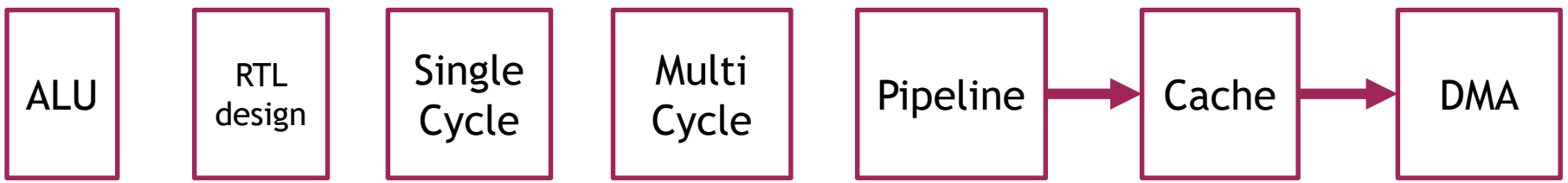
Lab05 Pipeline

Jeongmin Hong
jmhhh@postech.ac.kr

POSTECH
POHANG UNIVERSITY OF SCIENCE AND TECHNOLOGY

Lab Dependency

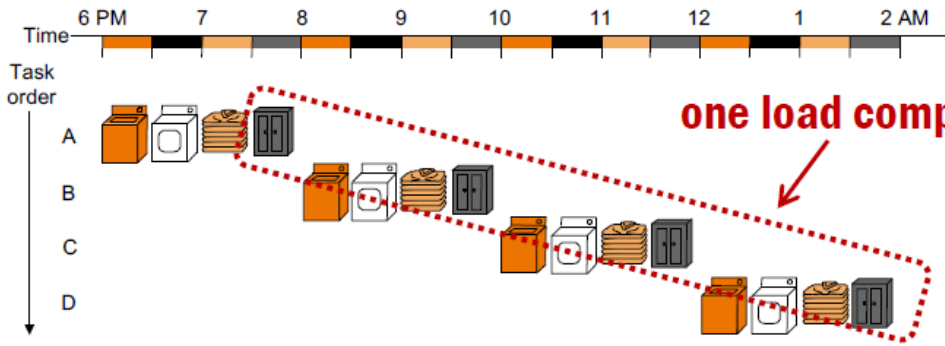


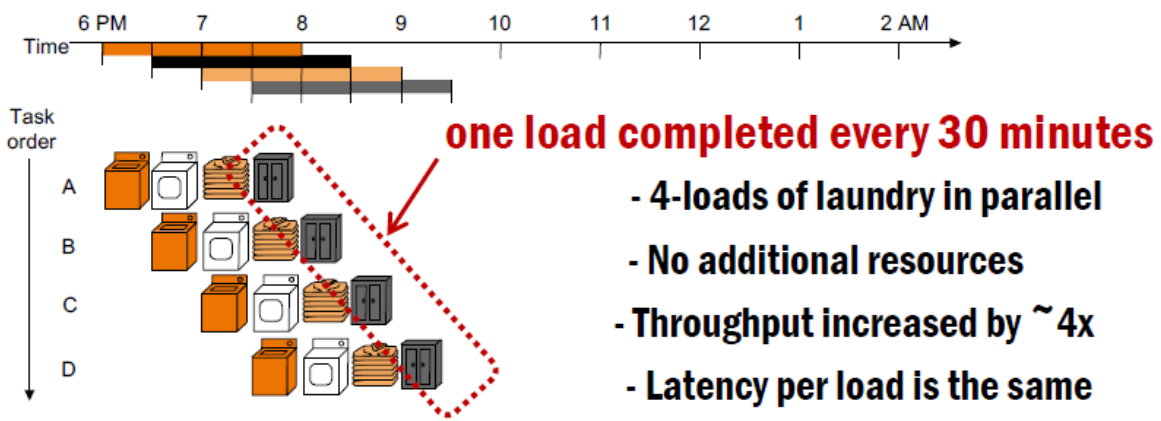
- From now on, you have to complete your assignment to start the next one
- Your implementation should be functionally correct
- So please start early! (This one will take a long time…)

Objective

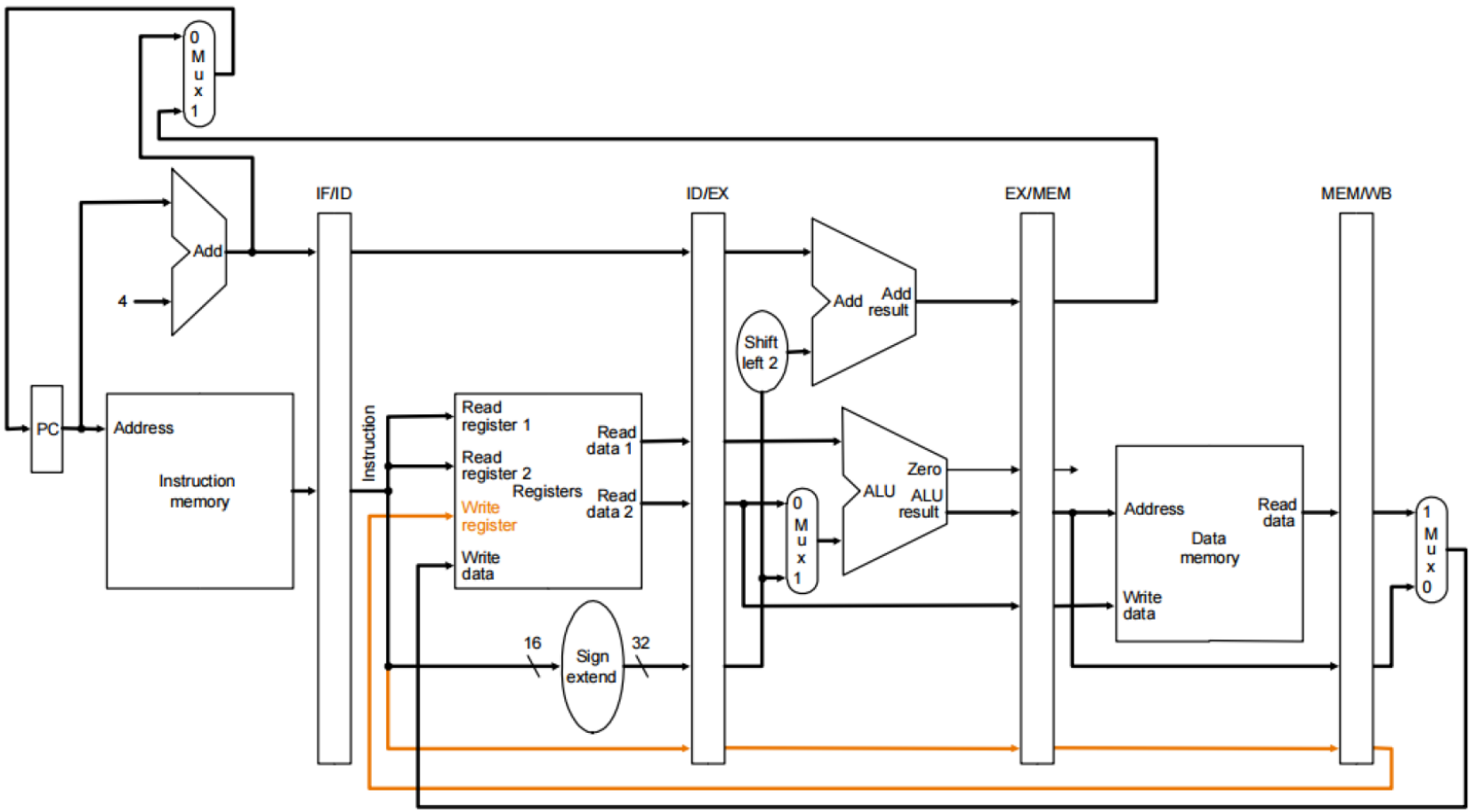
- To understand the reason why pipelined CPUs have better throughput
- To understand data & control hazards and how to solve them
- To design and implement the pipelined CPU

Why Do We Use Pipeline?

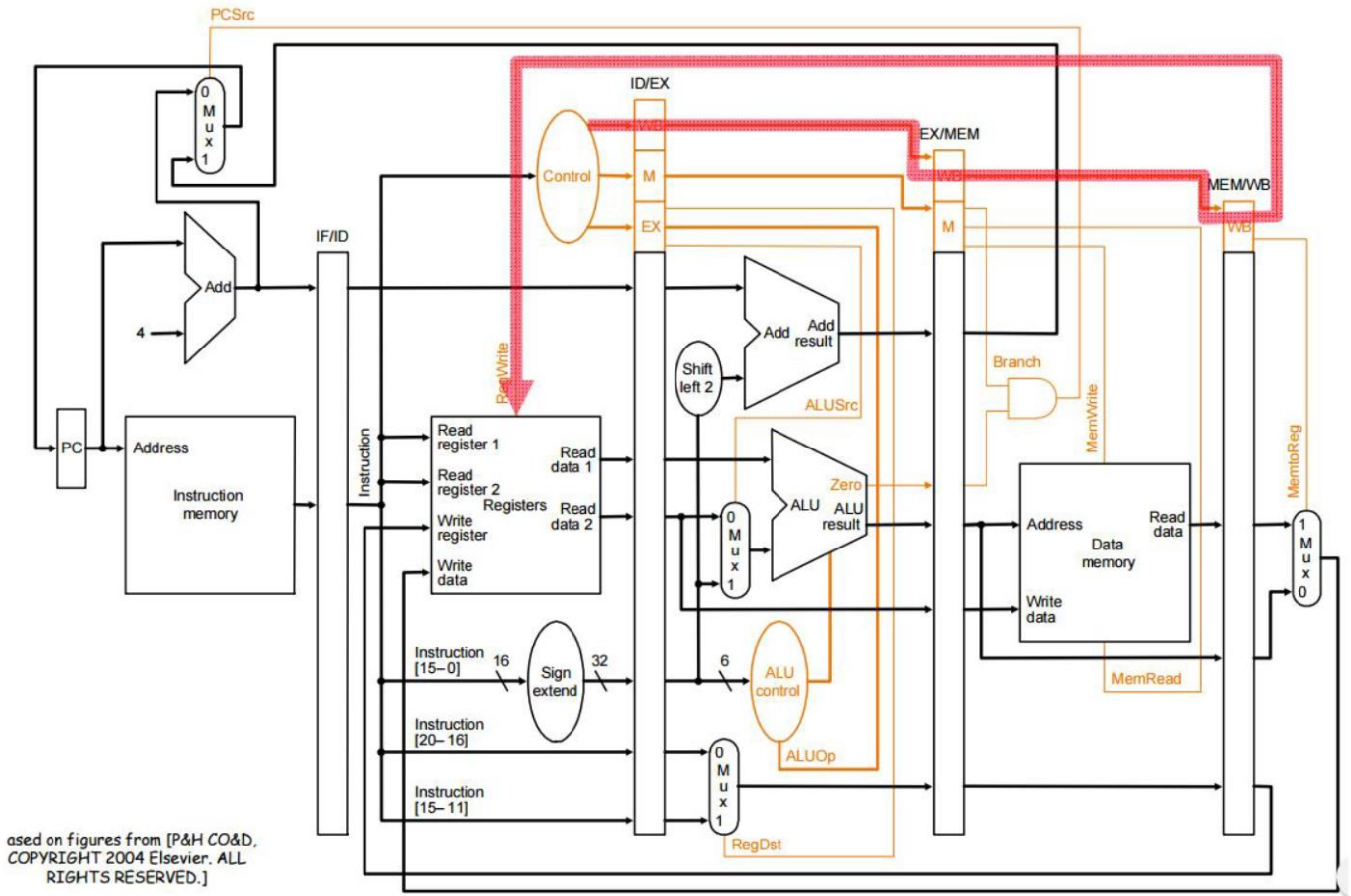




Pipeline: Datapath



Pipeline: Control

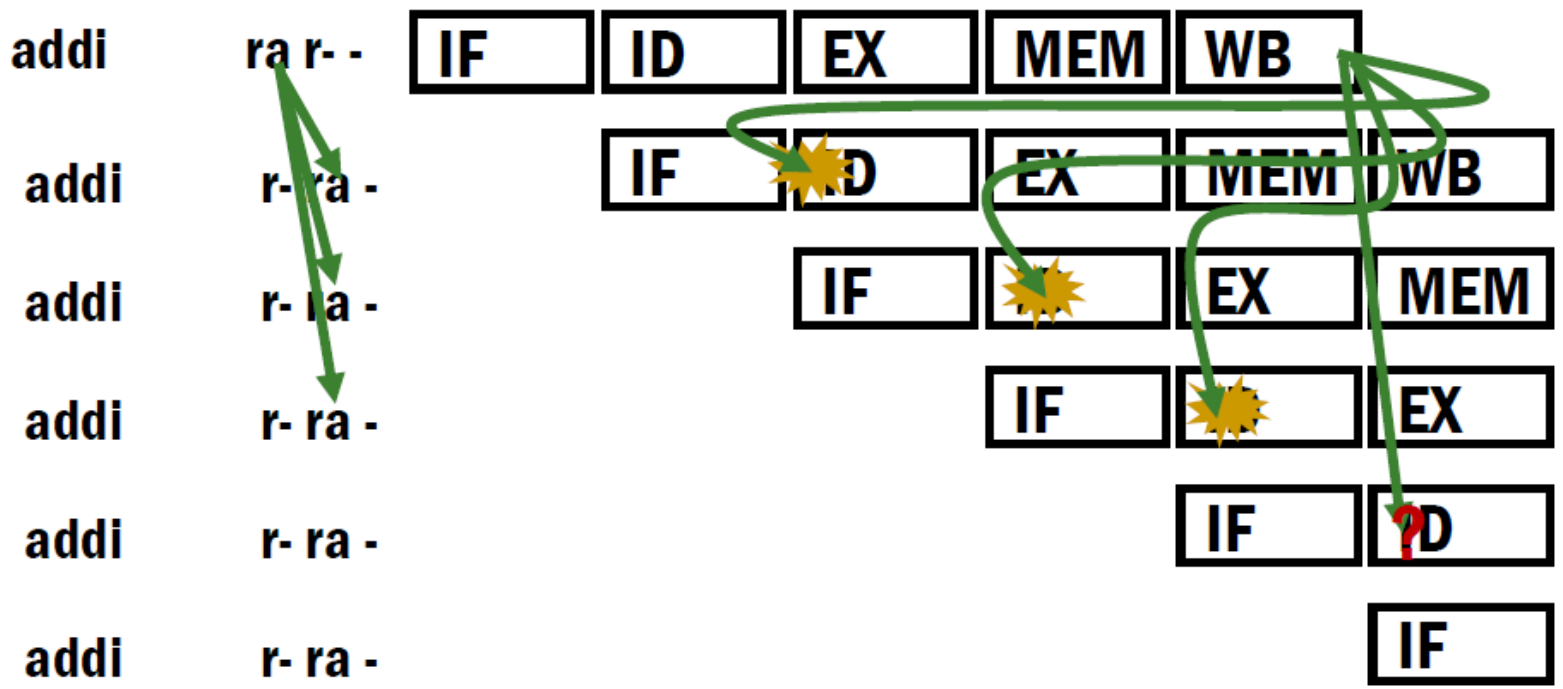


ased on figures from [P&H CO&D, COPYRIGHT 2004 Elsevier. ALL RIGHTS RESERVED.]

Hazards

- Pipeline hazards
  - Data hazard
  - Control hazard
  - Structural hazard

Data Hazard


addi ra r- -
addi r- ra -
addi r- ra -
addi r- ra -
addi r- ra -
addi r- ra -
IF
ID
EX
MEM
WB
IF
D
EX
MEM
WB
IF
EX
MEM
IF
EX
IF
D
IF

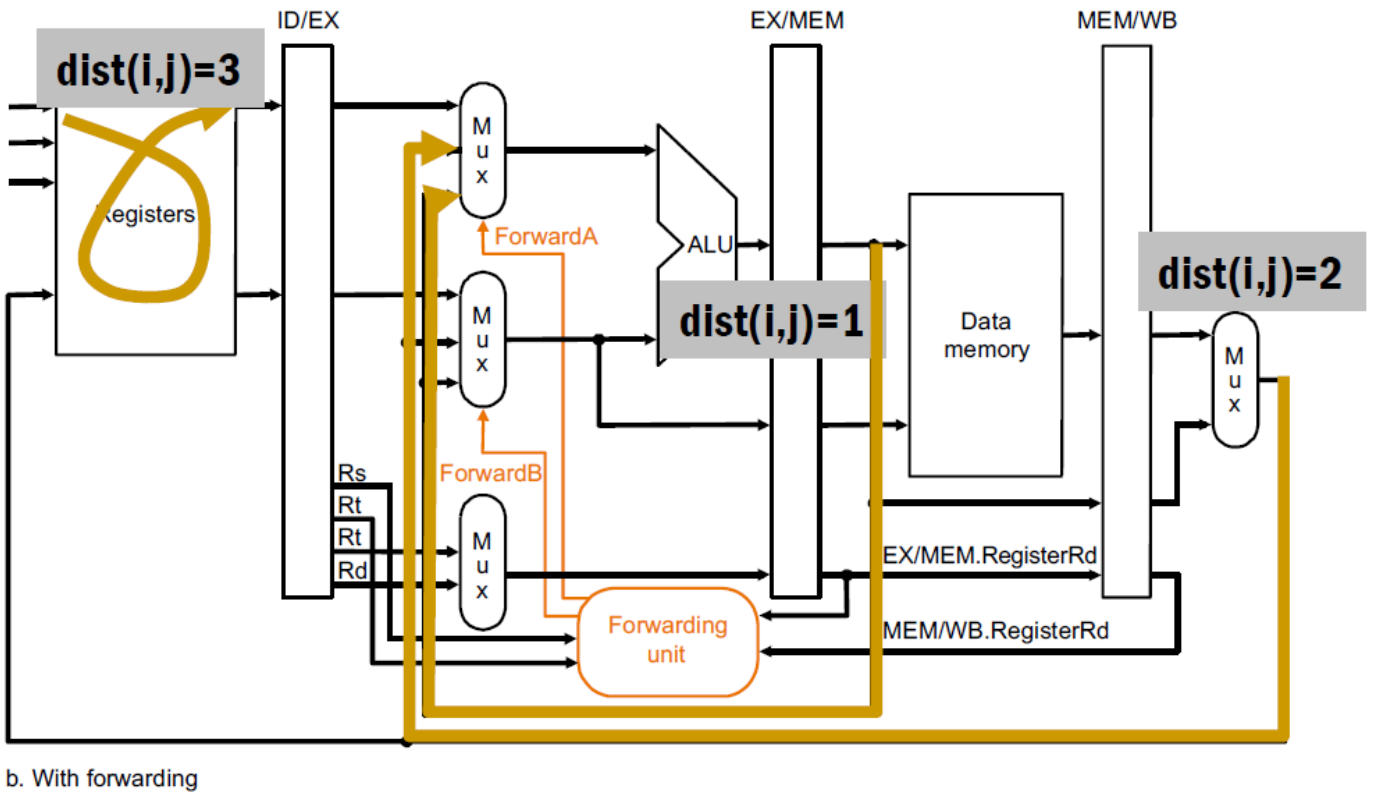
Data Hazard - Stall

	t_0	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}
IF	i	j	k	k	k	k	l				
ID	h	i	j	j	j	j	k	l			
EX		h	i	bub	bub	bub	j	k	l		
MEM			h	i	bub	bub	bub	j	k	l	
WB				h	i	bub	bub	bub	j	k	l

i: rx ← _

j: _ ← rx

Data Hazard – Forwarding (Extra Credit)



b. With forwarding

You will get extra credit (+5) for implementing data forwarding

Data Hazard

For this "updating register file in clock negative edge will be allowed!"

You will see why negedge is needed on register file update

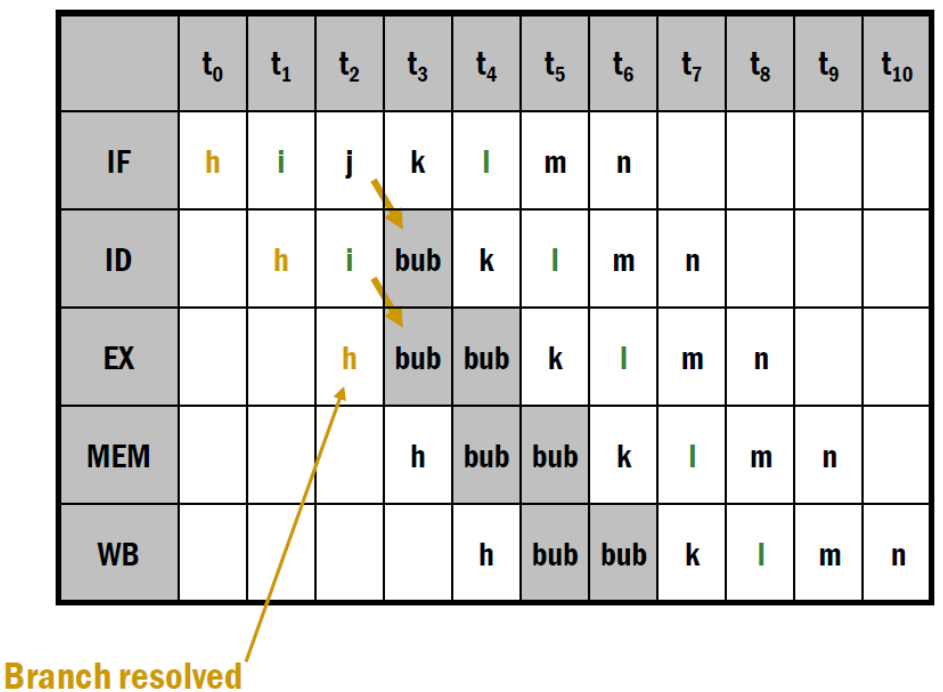
Control Hazard

	R/I-Type	LW	SW	Br	J	Jr
IF	use	use	use	use	use	use
ID	produce	produce	produce		produce	produce
EX				produce		
MEM						
WB						

- **PC hazard distance is at least 1**
- **Does that mean we must stall after every instruction?**
  - **IF stage can't know which PC to fetch next until the current PC is fetched and decoded**

Control Hazard - Flush

- You need to predict next PC as PC + 1
- Flush on misprediction
- Read textbook and lecture slides
- This is not optional!



Control Hazard – Branch Prediction

- Always not taken (no extra credit)
- Always taken (partial extra credit (+3) for branch prediction)
- 2-bit global prediction (full extra credit (+5) for branch prediction)

----------<Just for highly motivated students>----------

- Gshare
- Two-Level predictor
- Etc.

Requirements

- Implement the 5-stage pipelined CPU (baseline)
- Data Hazard (choose between stall or forwarding)
  - Stall (no extra credit)
  - Forwarding – distance 1,2 towards EXE stage & distance 3 internal forwarding (extra credit (+5))
- Control Hazard
  - Flush on misprediction (baseline)
  - Branch Prediction
    - Always not taken (no extra credit)
    - Always taken (partial extra credit (+3))
    - 2-bit global prediction (full extra credit (+5))
- You can modify skeleton code's module freely except cpu.v

Requirements for Report

- Compare the performance of multi-cycle CPU and pipeline CPU – write on report
- If you implement 2-bit global branch predictor for extra credit, You should compare the cycle count against CPU with always taken, always not taken predictor on your lab report
- If you implement always taken predictor for extra credit, You should compare the cycle count against CPU with always not taken predictor on your lab report
- Your implementation should be functionally correct!
  - The last test bench will be passed only if your implementation is functionally correct!