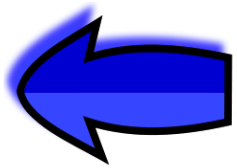
Pipelining

Slide courtesy: Smruti Ranjan Sarangi

Slides adapted by: Dr Sparsh Mittal

Outline

* Overview of Pipelining
* A Pipelined Data Path



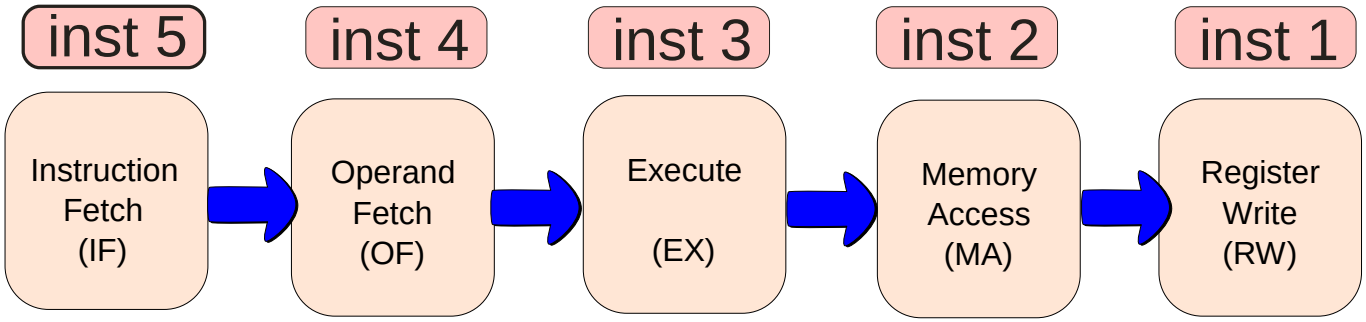
Designing Efficient Processors

- Without pipelining, there is lot of waste
  - We have 5 stages.
  - What is the IF stage doing, when the MA stage is active ?
    - **ANSWER** : It is **idling**

The Notion of Pipelining

- Let us understand the car assembly line
  - Is the engine shop idle, when the paint shop is painting a car ?
  - **NO** : It is building the engine of another car
  - When this engine goes to the body shop, it builds the engine of another car, and so on ....
  - **Insight** :
    - Multiple cars are built at the same time.
    - A car proceeds from one stage to the next

Pipelined Processors



* The IF, ID, EX, MA, and RW stages process 5 instructions simultaneously
* Each instruction proceeds from one stage to the next
* This is known as pipelining

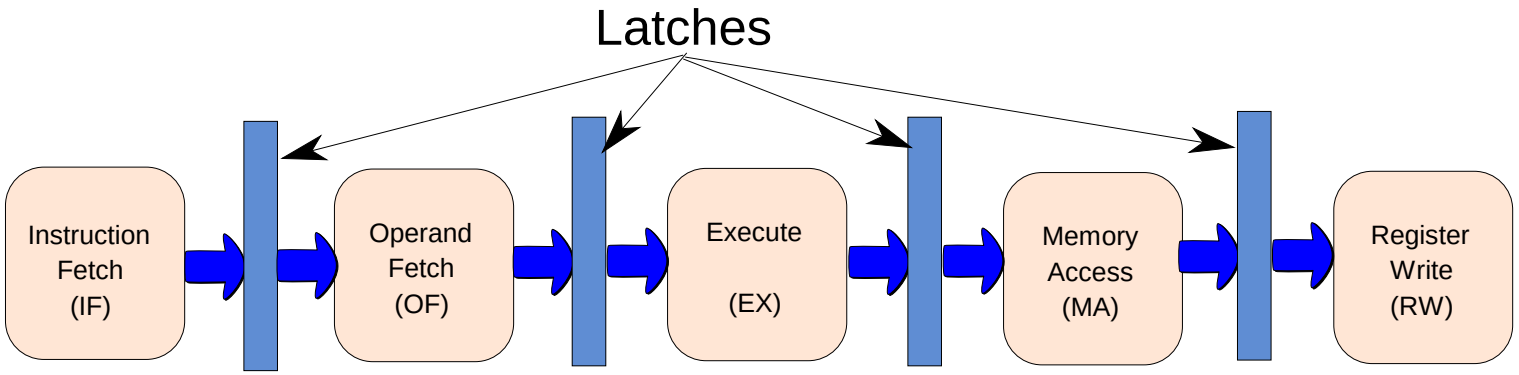
Advantages of Pipelining

- We keep all parts of the data path, busy all the time
- Let us assume that all the 5 stages do the same amount of **work**
  - Without pipelining, every T seconds, an instruction completes its execution
  - With pipelining, every T/5 seconds, a new instruction completes its execution

Design of a Pipeline

- Splitting the Data Path
  - We divide the data path into 5 parts : IF, OF, EX, MA, and RW
- Timing
  - We insert latches (registers) between consecutive stages
  - 4 Latches → IF-OF, OF-EX, EX-MA, and MA-RW
  - At the negative edge of a clock, an instruction moves from one stage to the next

Pipelined Data Path with Latches



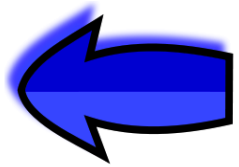
* Add a latch between subsequent stages.
  * Triggered by a negative clock edge

The Instruction Packet

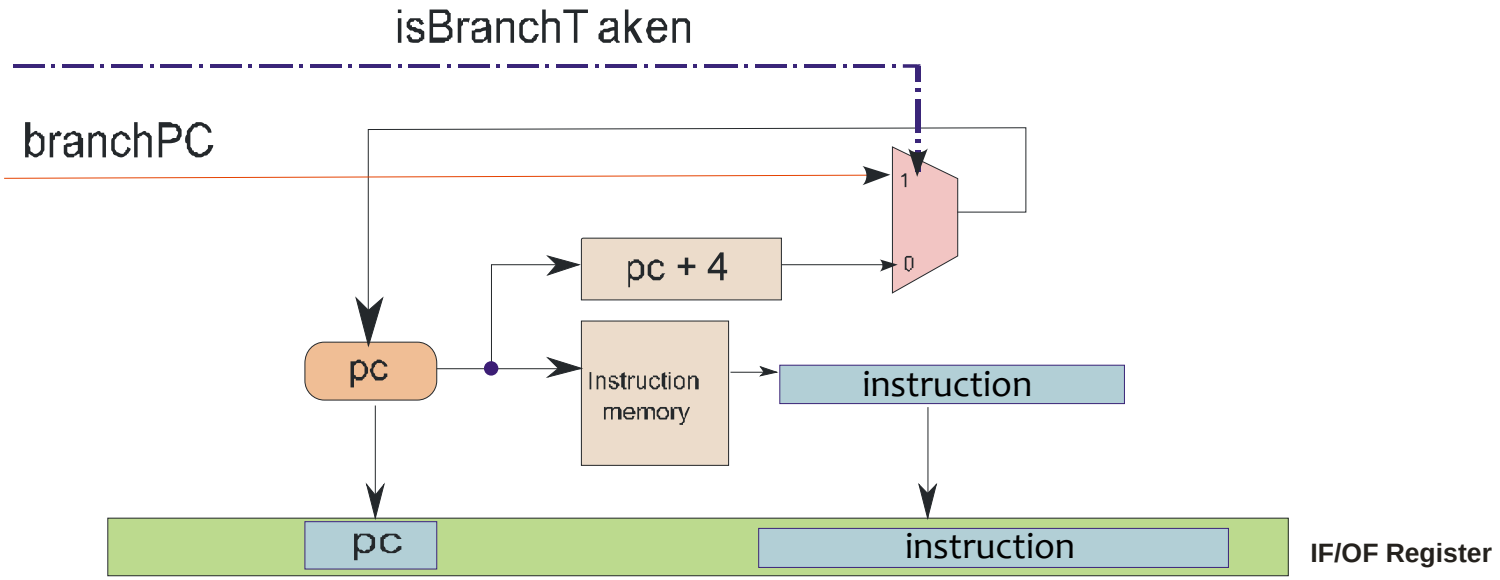
- What travels between stages ?
  - **ANSWER** : the instruction packet
- Instruction Packet
  - Instruction contents
  - Program counter
  - All intermediate results
  - Control signals
- Every instruction moves with its entire state, no interference between instructions

Outline

* Overview of Pipelining
* A Pipelined Data Path

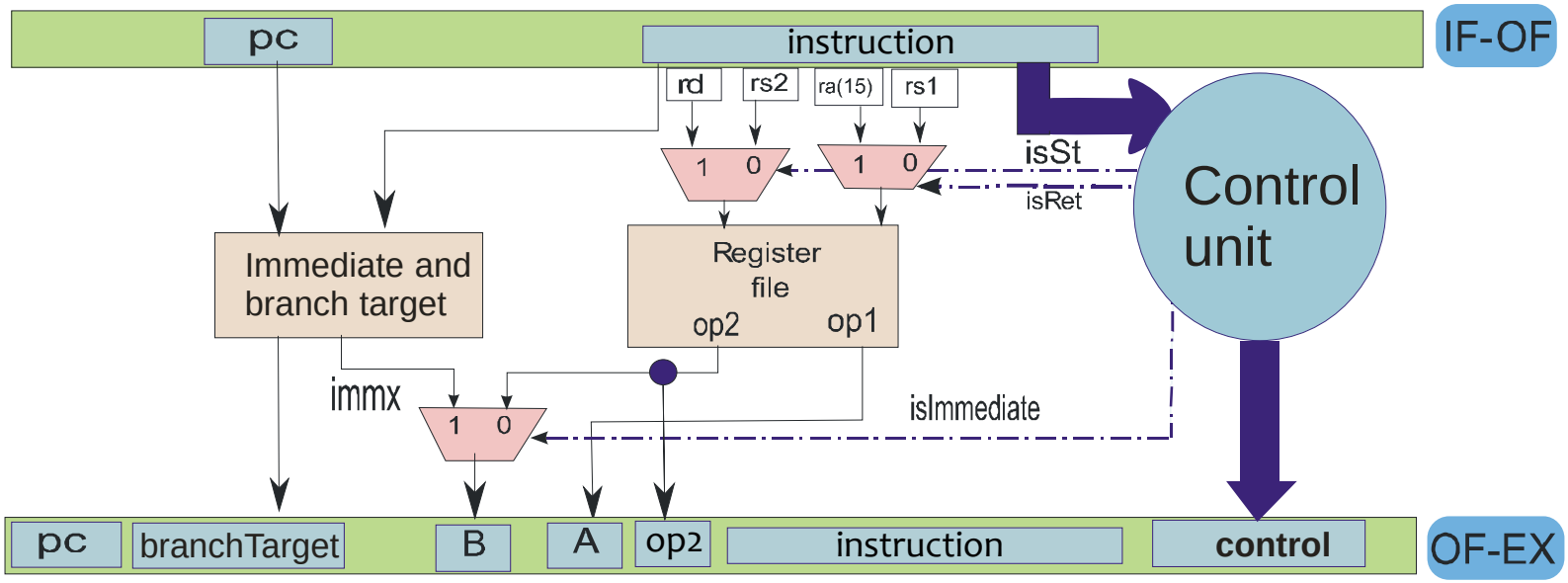


IF Stage



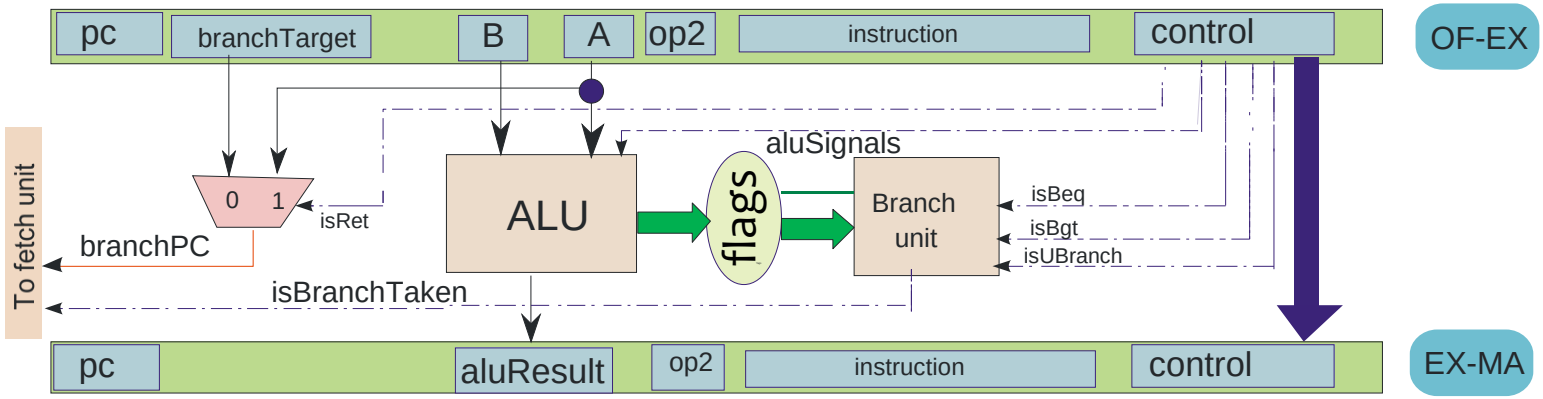
* Instruction contents saved in the instruction field

OF Stage



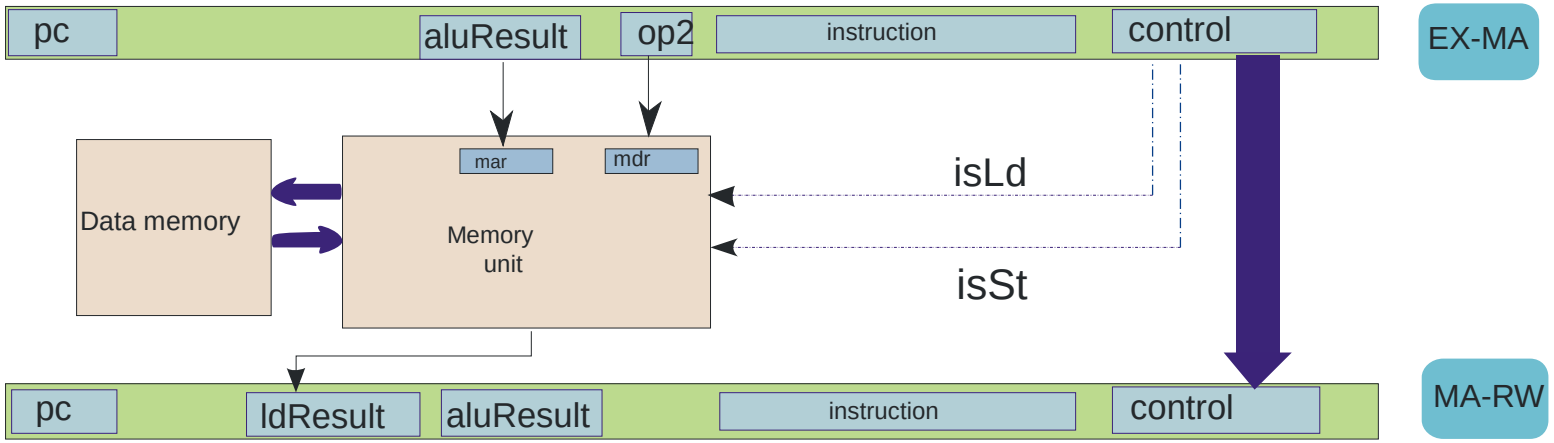
* A, B → ALU Operands, op2 (store operand), control (set of all control signals)

EX Stage



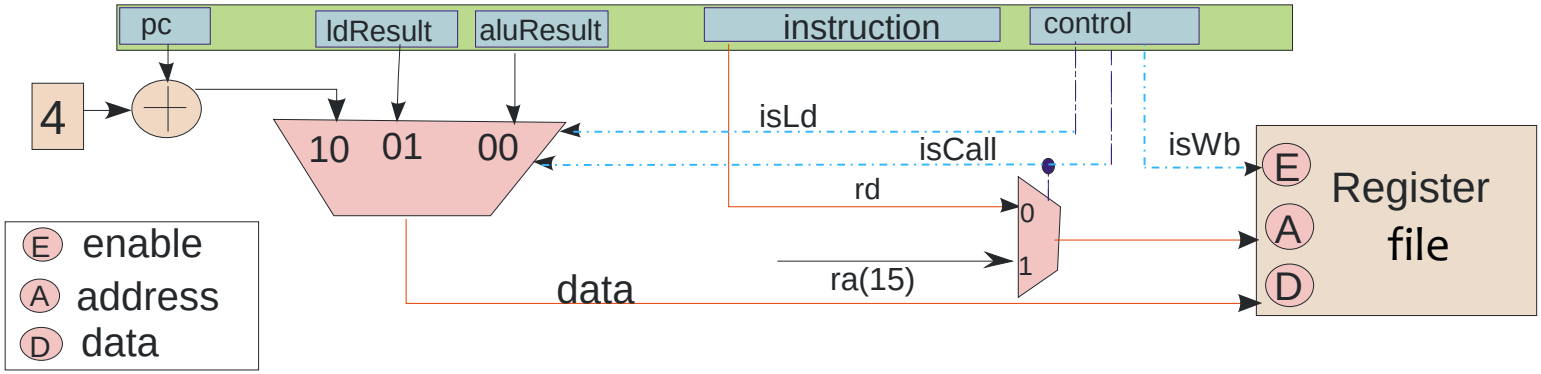
* aluResult → result of the ALU Operation
* op2, control, pc, instruction (passed from OF-EX)

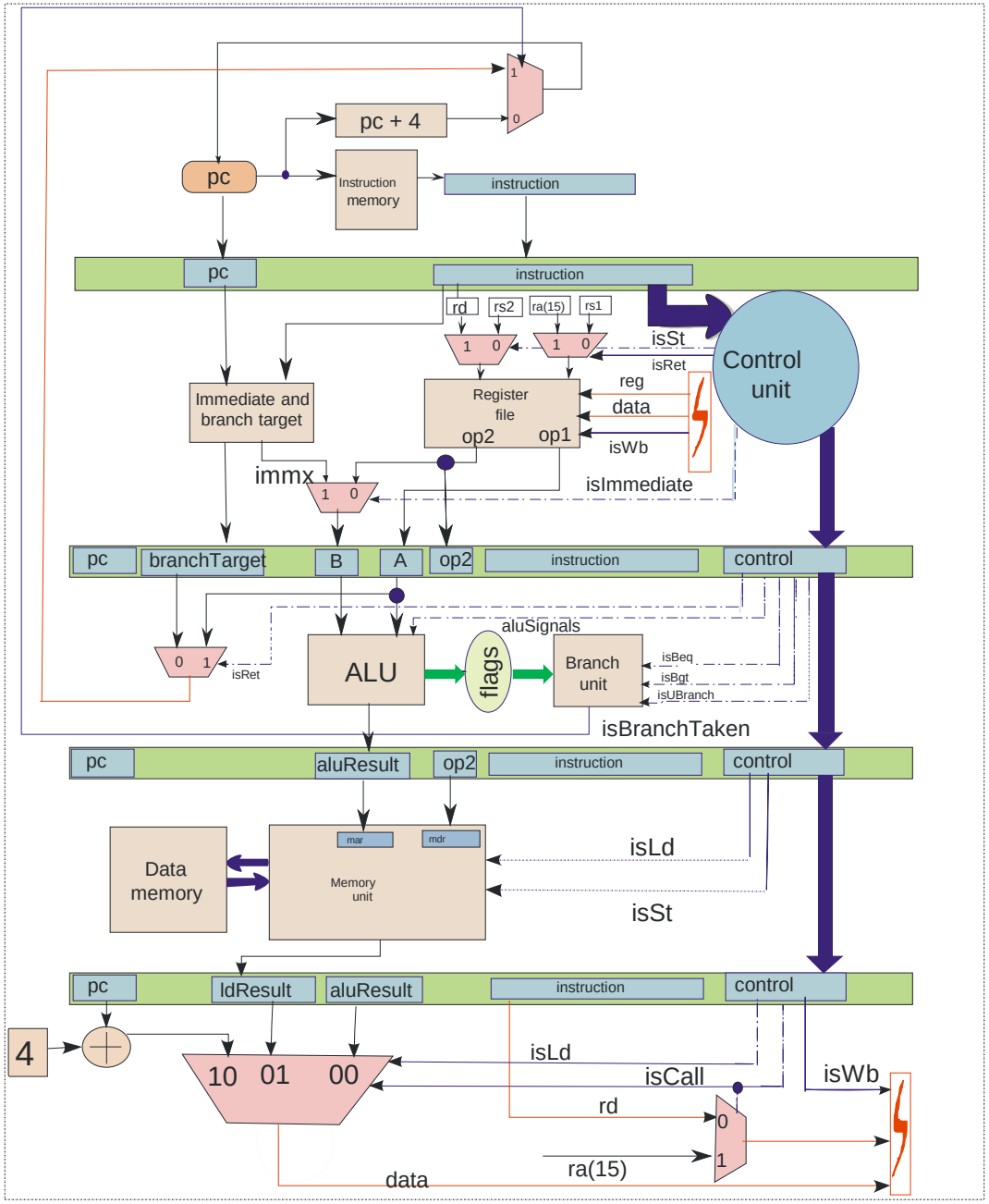
MA Stage



* ldResult → result of the load operation
* aluResult, control, pc, instruction (passed from EX-MA)

RW Stage


pc
ldResult
aluResult
instruction
control
4
10
01
00
isLd
isCall
isWb
rd
0
1
ra(15)
data
E
A
D
Register file
E enable
A address
D data


1
0
pc + 4
pc
Instruction memory
instruction
pc
instruction
rd
rs2
ra(15)
rs1
1 0
1 0
isSt
isRet
Control unit
reg
data
isWb
Immediate and branch target
Register file
op2
op1
immx
1 0
isImmediate
pc
branchTarget
B
A
op2
instruction
control
aluSignals
0 1
isRet
ALU
flags
Branch unit
isBeq
isBgt
isUBranch
isBranchTaken
pc
aluResult
op2
instruction
control
Data memory
mar
mdr
Memory unit
isLd
isSt
pc
ldResult
aluResult
instruction
control
4
10 01 00
isLd
isCall
isWb
rd
0
1
ra(15)
data

Abridged Diagram