ICS 233
COMPUTER ARCHITECTURE

Pipelined Processor Design

Enhancing Performance with Pipelining

Lecture 24

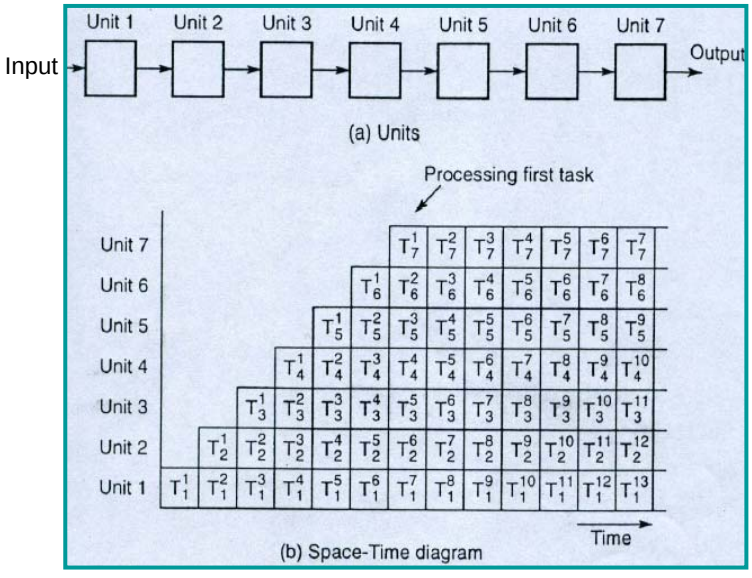
Pipelining

- **A very effective way of achieving parallelism within the processor is by the use of pipelining**
  - In pipelining, each task or operation is divided into a number of subtasks that need to be performed in sequence
  - Each subtask is performed by its own logical unit, rather than by a single unit which performs all the subtasks.
  - The units are connected together in a serial fashion with the output of one connecting to the input of the next, and all the units operate simultaneously.
  - While one unit is performing a subtask on the ith task, the preceding unit in the pipeline is performing its subtask on the (i+1)th task.

Pipelining

- **Pipelining is an implementation technique in which multiple instructions are overlapped in execution.**
- **Pipelining is key to making processors fast.**

Pipelining



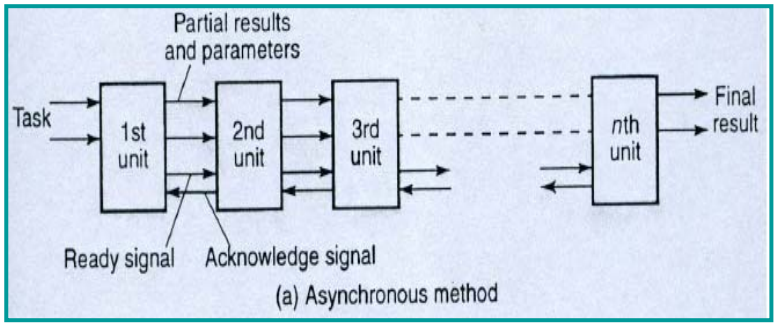
Pipeline Data Transfer

- **Two methods of implementing the information transfer in a pipeline**
  - **Asynchronous method**
  - **Synchronous method**

Asynchronous Pipeline Data Transfer

- In the asynchronous method, a pair of "handshaking" signals are used between each unit and the next unit
  - a ready signal
  - an acknowledge signal
- The ready signal informs the next unit that it has finished the present operation and is ready to pass the task and any results onwards
- The acknowledge signal is returned when the receiving unit has accepted the task and results.

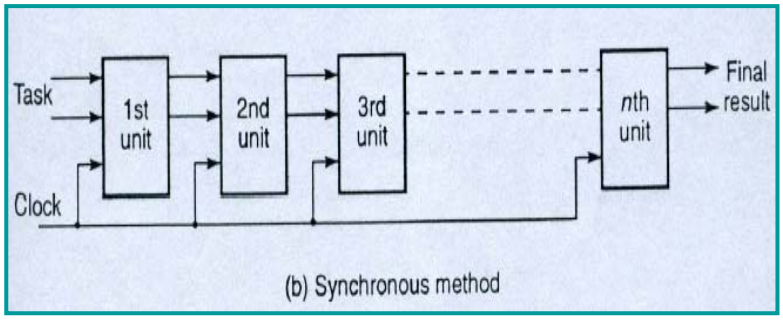
Asynchronous Pipeline Data Transfer



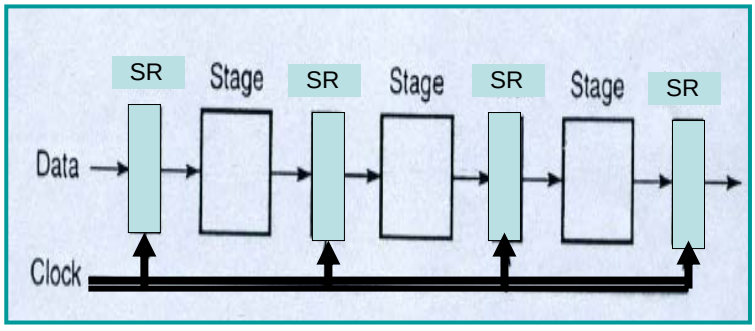
Synchronous Pipeline Data Transfer

- In the Synchronous method, one timing signal causes all outputs of units to be transferred to the succeeding units
- The timing signal occurs at fixed intervals, taking into account the slowest unit.
- Instruction and arithmetic pipelines use the synchronous method
- In the synchronous method, there is a staging register between each unit and the clock signal activates all the staging registers simultaneously.
- Staging registers are used between the stages to hold the information.

Synchronous Pipeline Data Transfer



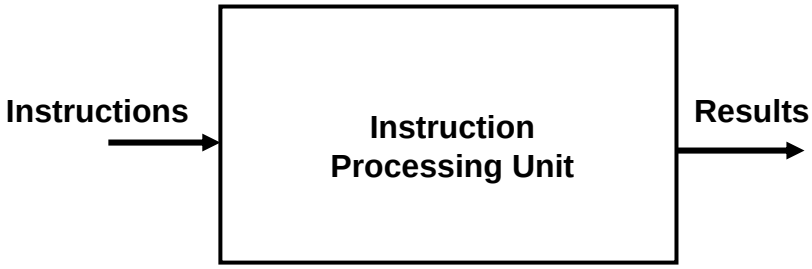
Synchronous Pipeline with staging registers



Pipelining

- **Pipelining improves performance by increasing instruction throughput, as opposed to decreasing the execution time of an individual instruction**
- **Instruction throughput is the important metric because real programs execute billions of instructions**

Instruction Processing Unit (IPU)



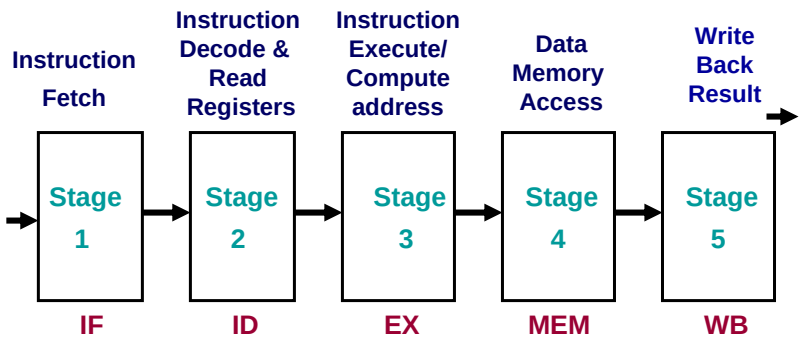
MIPS instructions classically take five steps

Instruction Processing Unit

- **Steps performed by Instruction Processing Unit are :**
  - **Instruction Fetch**
    - **Fetch instructions from memory**
  - **Instruction Decode & Read Registers**
    - **Read registers while decoding the instruction**
  - **Instruction Execute/ Compute Address**
    - **Execute the operation or calculate an address**
  - **Data Memory Access**
    - **Access an operand in data memory**
  - **Write Back Result**
    - **Write the result into a register**

Instruction Processing Unit

- **5-Stage Pipelined Instruction Processing Unit**

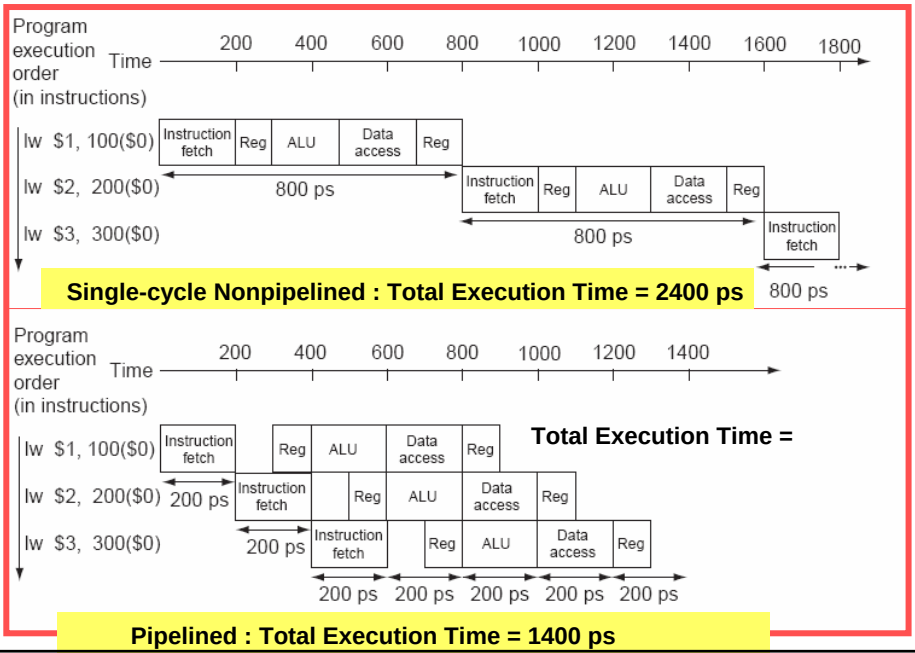


Instruction Processing Unit

➢ **Example : Total Execution time using Single-cycle Non-pipelined Vs. Pipelined processor**

Instruction Class	IF	ID	EX	MEM	WB	Total Time
Load Word (lw)	200 ps	100 ps	200 ps	200 ps	100 ps	800 ps
Store Word (sw)	200 ps	100 ps	200 ps	200 ps		700 ps
R-format (and, sub, and or, slt)	200 ps	100 ps	200 ps		100 ps	600 ps
Branch (Beq)	200 ps	100 ps	200 ps			500 ps

Total Execution time Non-pipelined Vs. Pipelined



Instruction Processing Unit

➢ **MIPS – Designing Instruction Sets for Pipelining**

- **First, all MIPS instructions are the same length**

  - This restriction makes it easier to fetch instructions in the first pipeline stage and to decode them in the second stage

- **Second, MIPS has only a few instruction formats, with the source register fields being located in the same place in each instruction.**

  - This symmetry means that the second stage can begin reading the register file at the same time that the hardware is determining what type of instruction was fetched.

- **Third, memory operands only appear in loads or stores in MIPS.**

  - This restriction means that the execute stage can be used to calculate the memory address and then access memory in the following stage.

- **Fourth, operands must be aligned in memory**

  - The requested data can be transferred between processor and memory in a single pipeline stage.

IPU - Pipelined Datapath

➢ **Separate the datapath into five stages :**

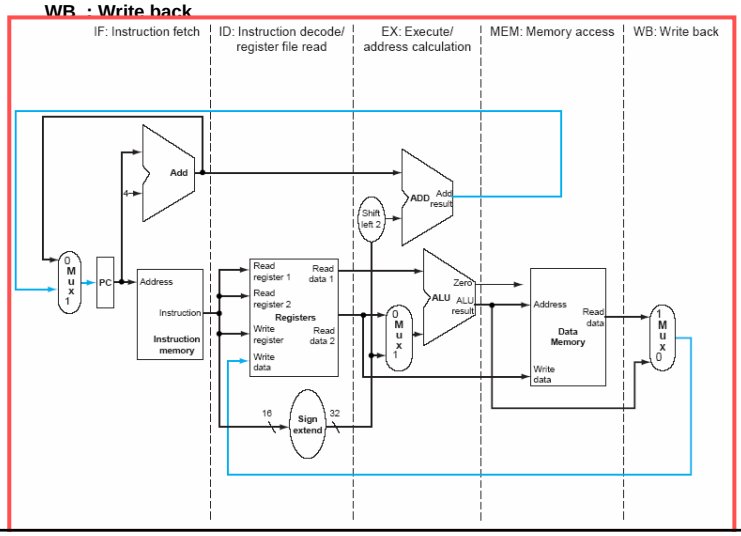
IF: Instruction Fetch,

ID : Instruction decode and Register file read

EX : Execution or Address Calculation

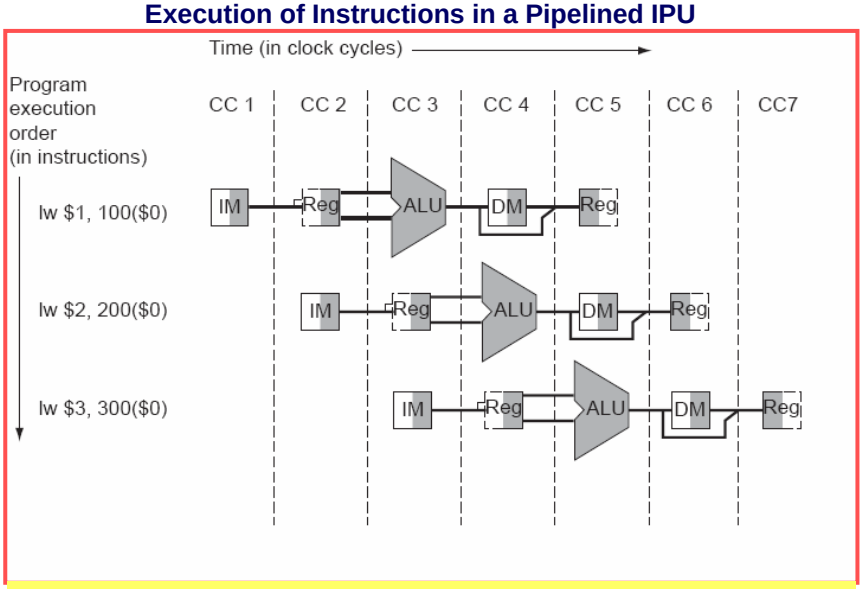
MEM : Data Memory Access

WB : Write back



IPU - Pipelined Datapath

Execution of Instructions in a Pipelined IPU

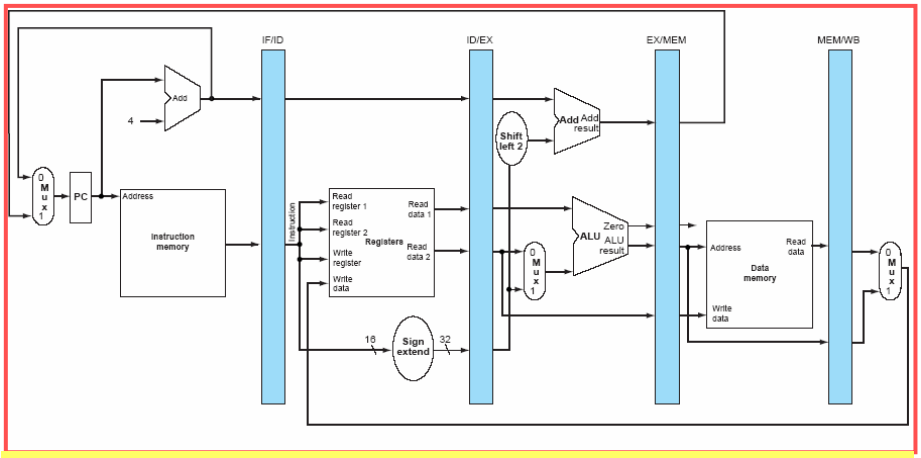


Multiple-clock-cycle pipeline diagram of three instructions

19

IPU - Pipelined Datapath

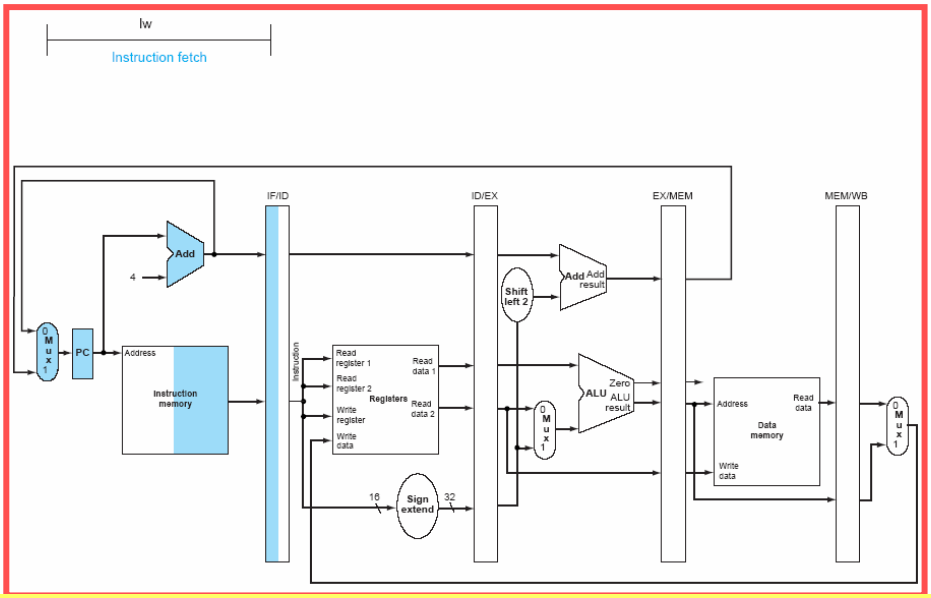
Pipelined version of the IPU Datapath



Four Pipeline Registers are : IF/ID, ID/EX, EX/MEM and MEM/WB

lw instruction execution in the Pipelined IPU – Clock Cycle 1

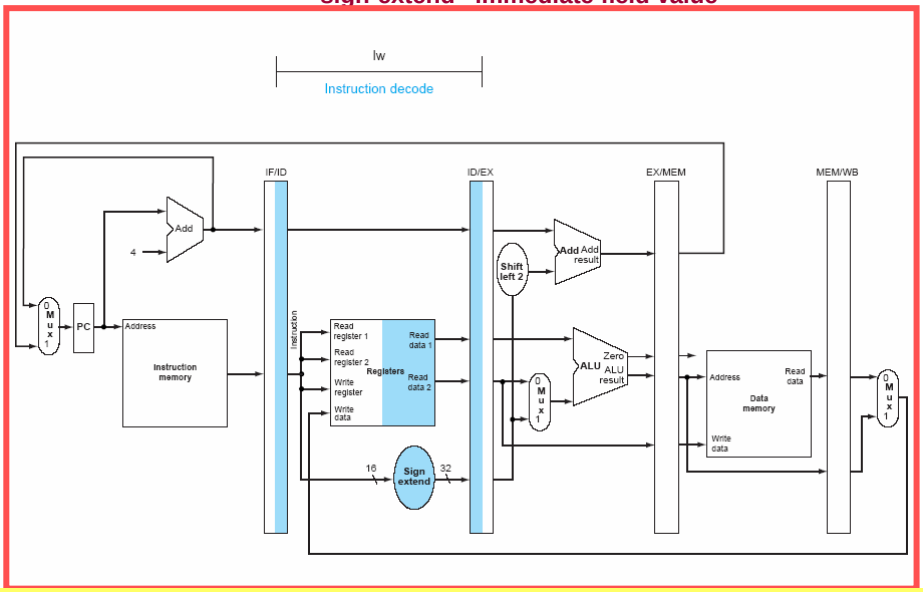
Pipe Stage 1(IF Stage) : Fetching of instruction lw, update PC (PC = PC +4)



Single-cycle pipeline diagram for Clock Cycle 1

lw instruction execution in the Pipelined IPU – Clock Cycle 2

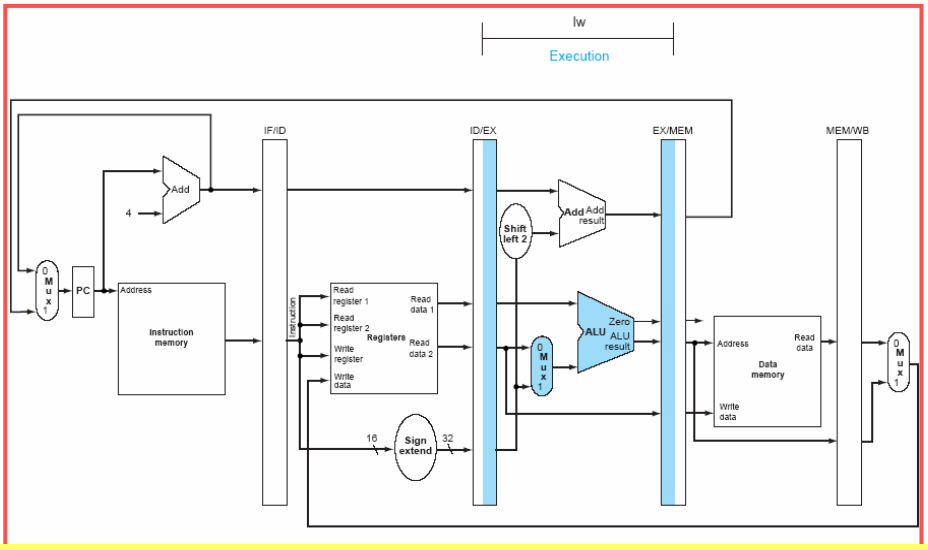
Pipe stage 2 (ID Stage) : Decoding of instruction lw, Register Read, sign-extend immediate field value



Single-cycle pipeline diagram for Clock Cycle 2

lw instruction execution in the Pipelined IPU – Clock Cycle 3

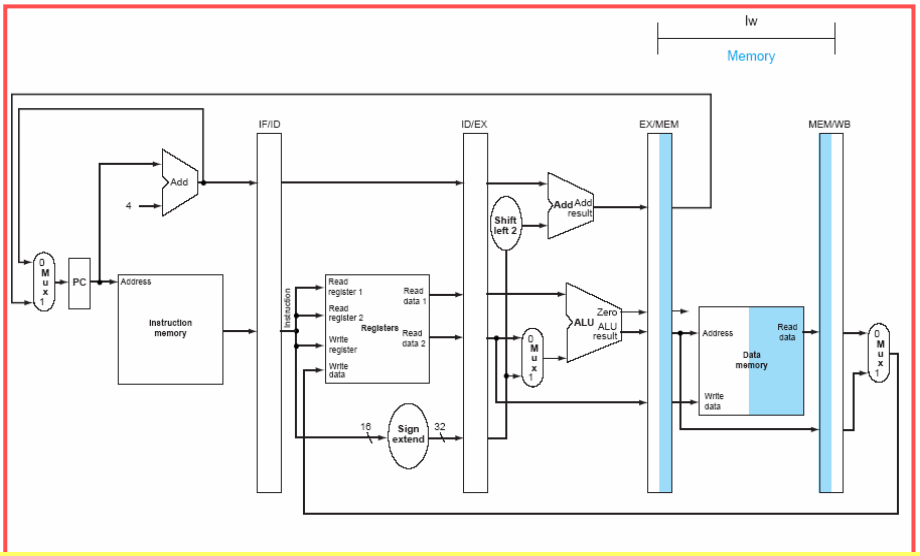
Pipe stage 3 (EX Stage) : lw instruction memory address computation



Single-cycle pipeline diagram for Clock Cycle 3

lw instruction execution in the Pipelined IPU – Clock Cycle 4

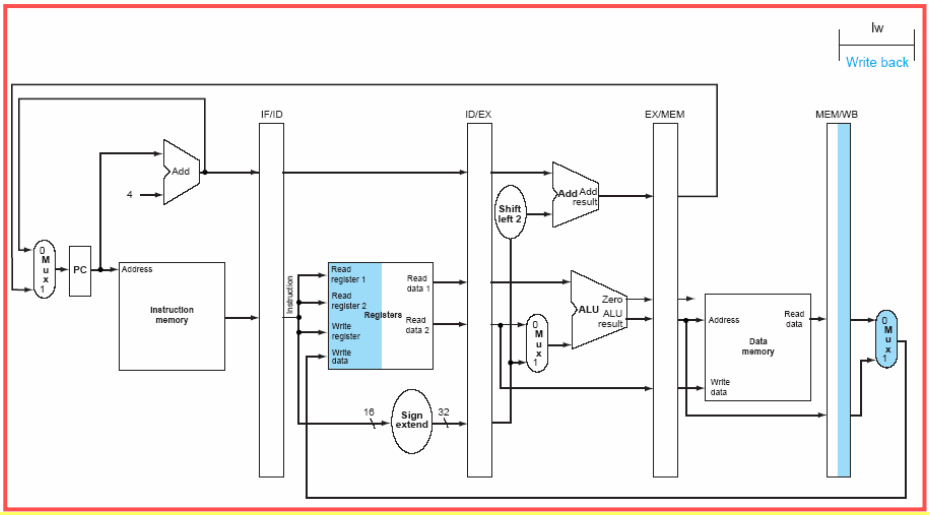
Pipe stage 4 (MEM Stage) : Data Memory Read for lw instruction



Single-cycle pipeline diagram for Clock Cycle 4

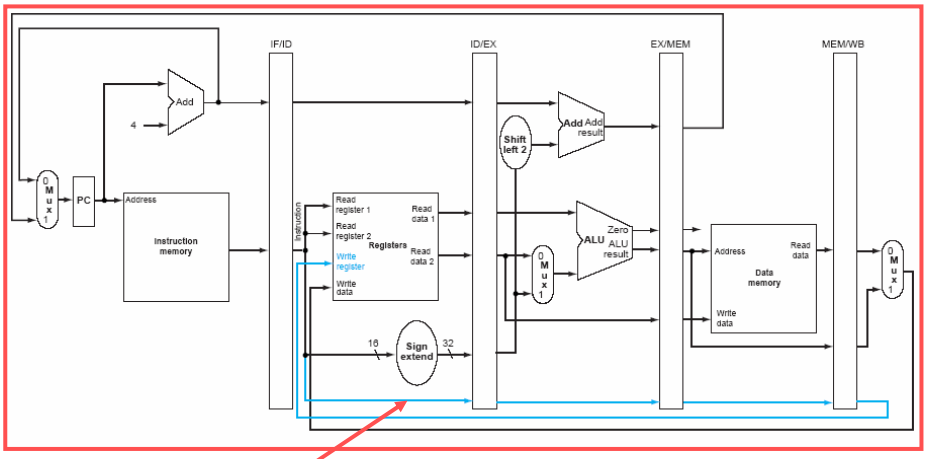
lw instruction execution in the Pipelined IPU – Clock Cycle 5

Pipe stage 5 (WB Stage) : Write data to Register from Memory for lw instruction



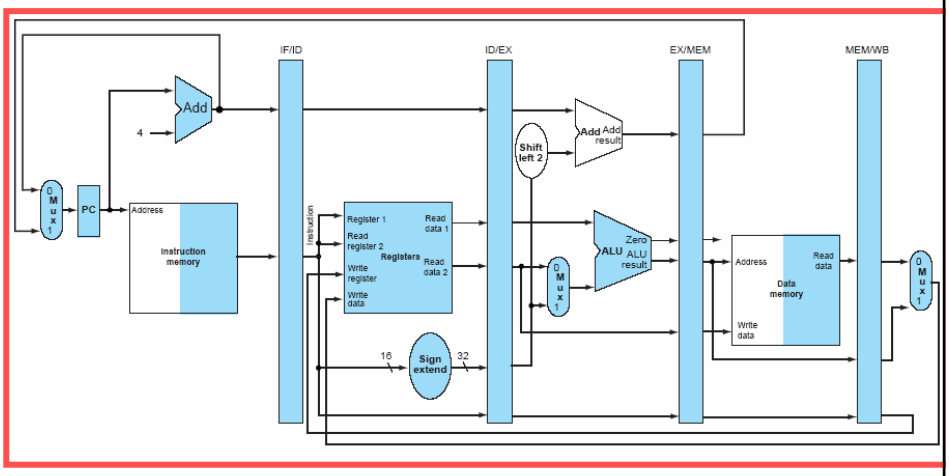
Single-cycle pipeline diagram for Clock Cycle 5

Modified Pipelined IPU for lw instruction execution



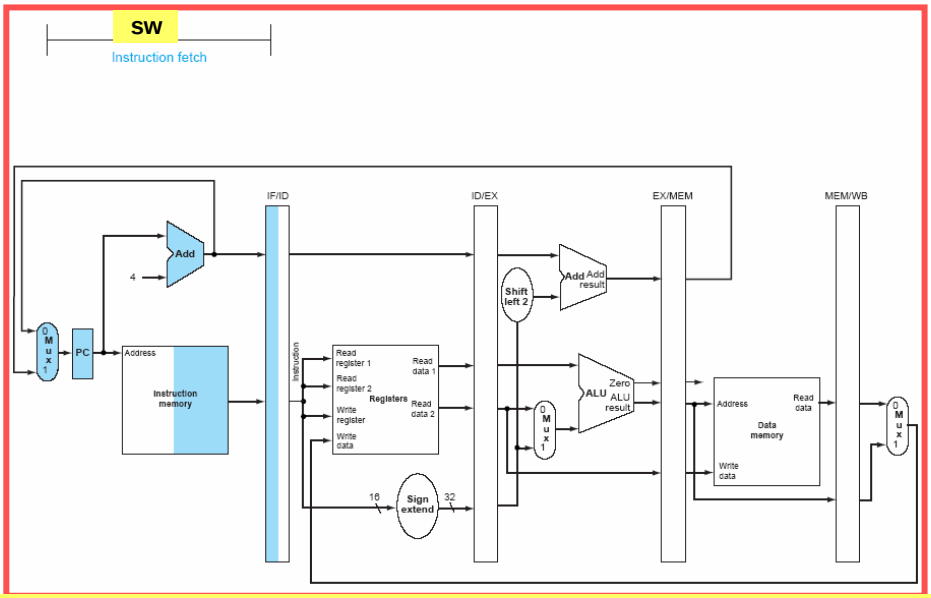
Preserve the destination register number for use in the WB stage

Portion of the Pipelined IPU datapath used in all five stages of a lw instruction execution



sw instruction execution in the Pipelined IPU – Clock Cycle 1

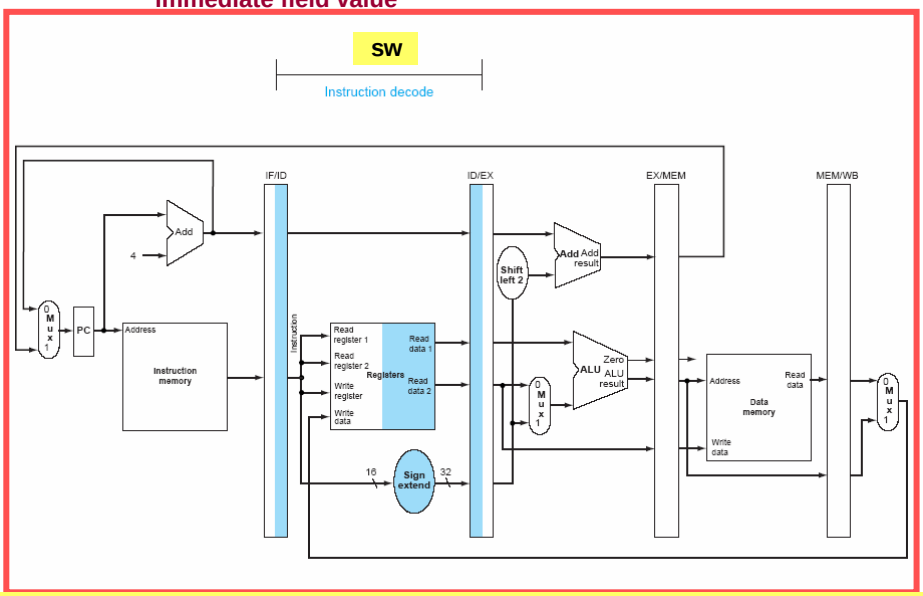
IF Stage : Fetching of instruction sw, update PC (PC = PC +4)



Single-cycle pipeline diagram for Clock Cycle 1

sw instruction execution in the Pipelined IPU – Clock Cycle 2

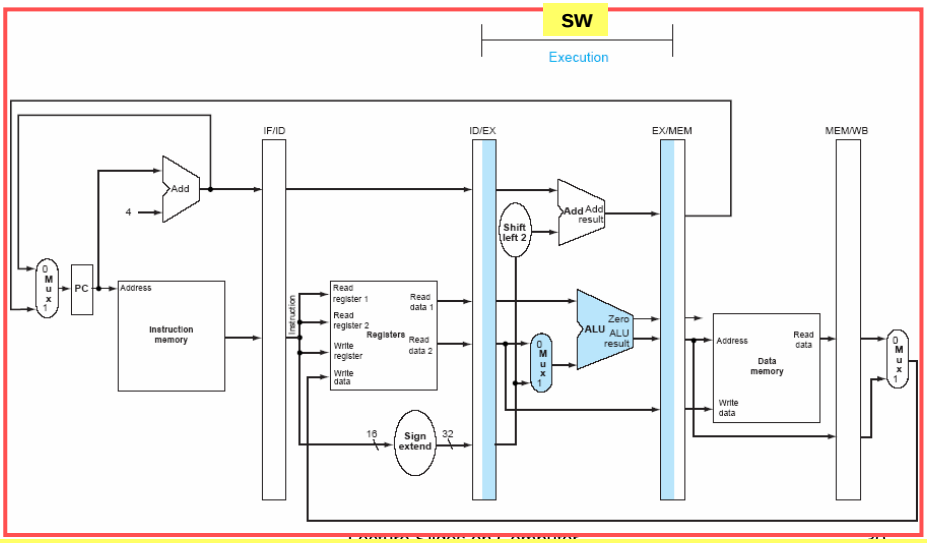
ID Stage : Decoding of instruction sw, Register Read, sign-extend immediate field value



Single-cycle pipeline diagram for Clock Cycle 2

sw instruction execution in the Pipelined IPU – Clock Cycle 3

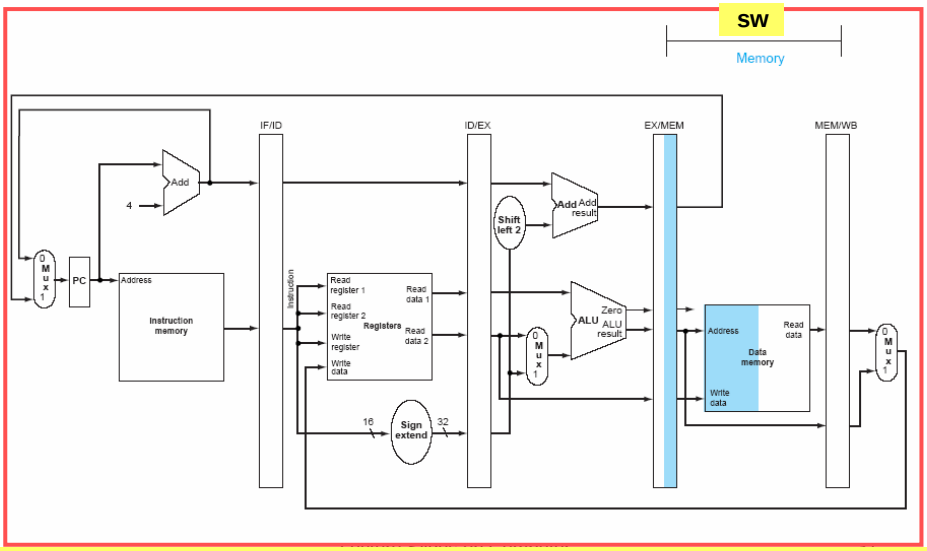
EX Stage : sw instruction memory address computation



Single-cycle pipeline diagram for Clock Cycle 3

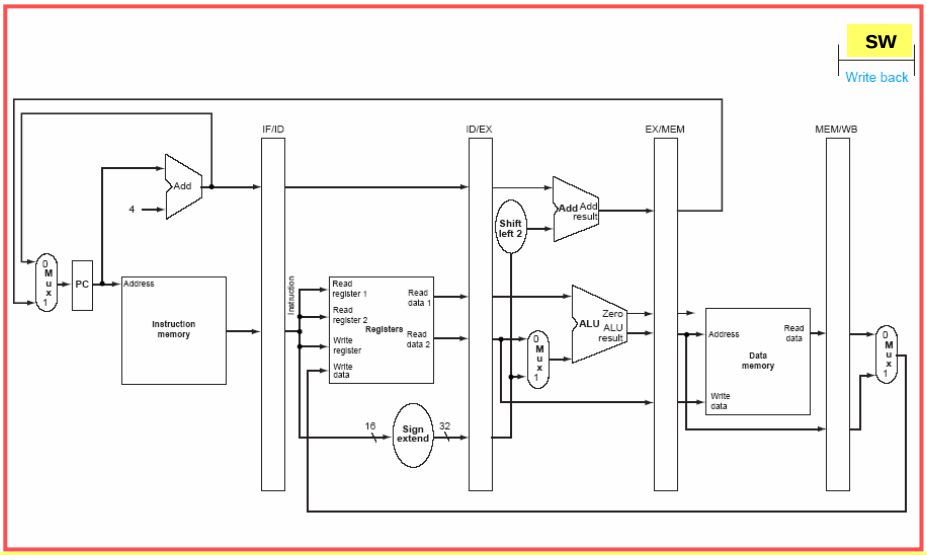
sw instruction execution in the Pipelined IPU – Clock Cycle 4

MEM stage : Data Memory write for sw instruction



Single-cycle pipeline diagram for Clock Cycle 4

sw instruction execution in the Pipelined IPU – Clock Cycle 5



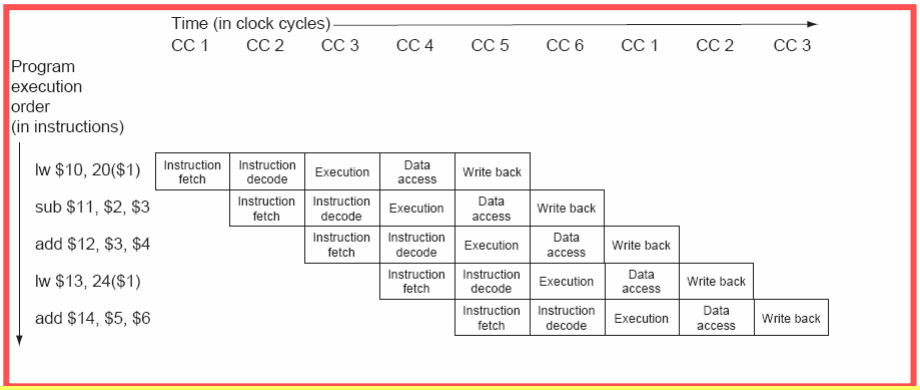
Nothing happens in WB Stage

Single-cycle pipeline diagram for Clock Cycle 5

Graphical Representation of Pipelines

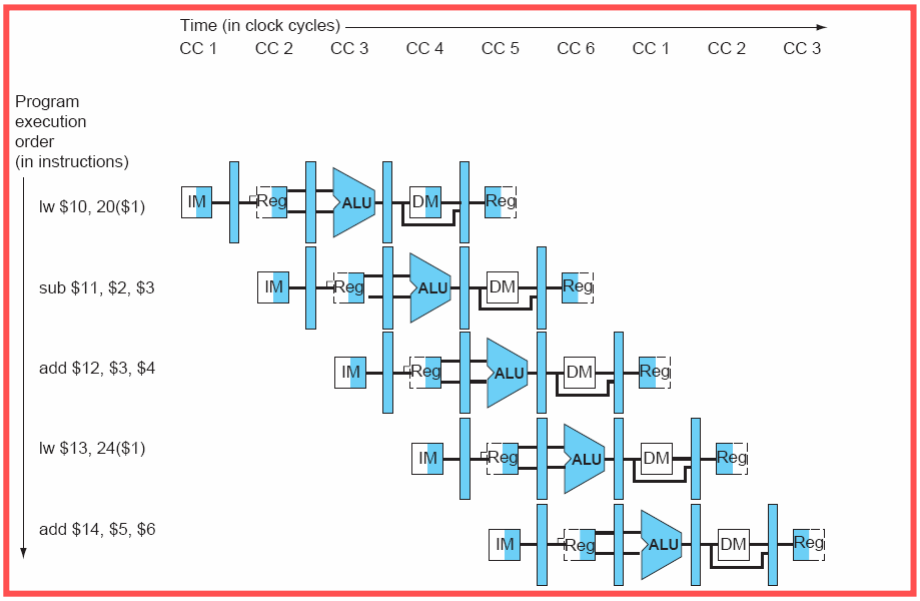
- **Consider the execution of following the code sequence using the 5-stage pipeline**

```
lw    $10, 20($1)
sub   $11, $2, $3
add   $12, $3, $4
lw    $13, 24($1)
add   $14, $5, $6
```



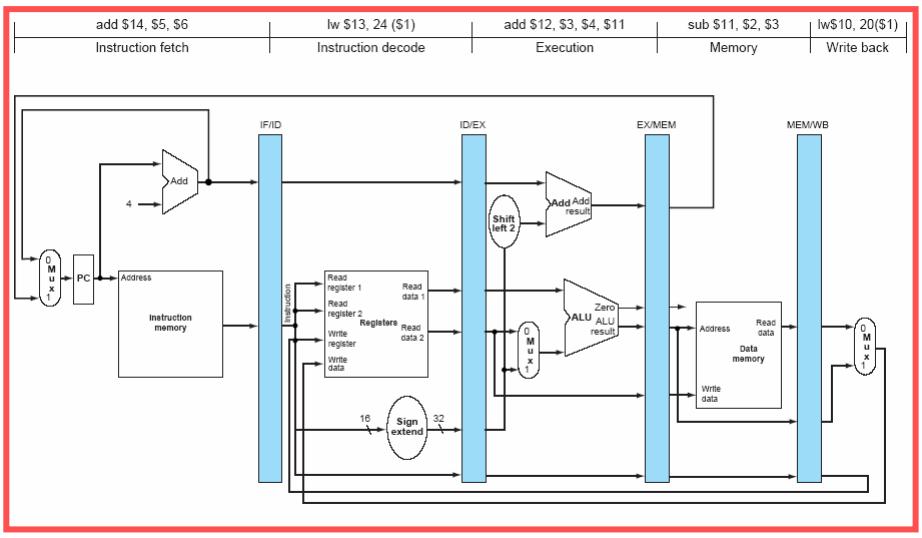
Traditional Multiple-clock cycle pipeline diagram of five instructions

Graphical Representation of Pipelines



Multiple-clock cycle pipeline diagram of five instructions

Graphical Representation of Pipelines



Single clock cycle pipeline diagram of five instructions in execution – Clock Cycle 5