4.4 PIPELINING

Pipelining is an implementation technique in which multiple instructions are overlapped in execution. The computer pipeline is divided in stages. Each stage completes a part of an instruction in parallel. The stages are connected one to thenext to form a pipe - instructions enter at one end, progress through the stages, and exit at the other end. Today,. The *non-pipelined* approach to laundry would be as follows:

1. Place one dirty load of clothes in the washer.
2. When the washer is finished, place the wet load in the dryer.
3. When the dryer is finished, place the dry load on a table and fold.
4. When folding is finished, ask your roommate to put the clothes away.When your roommate is done, start over with the next dirty load.

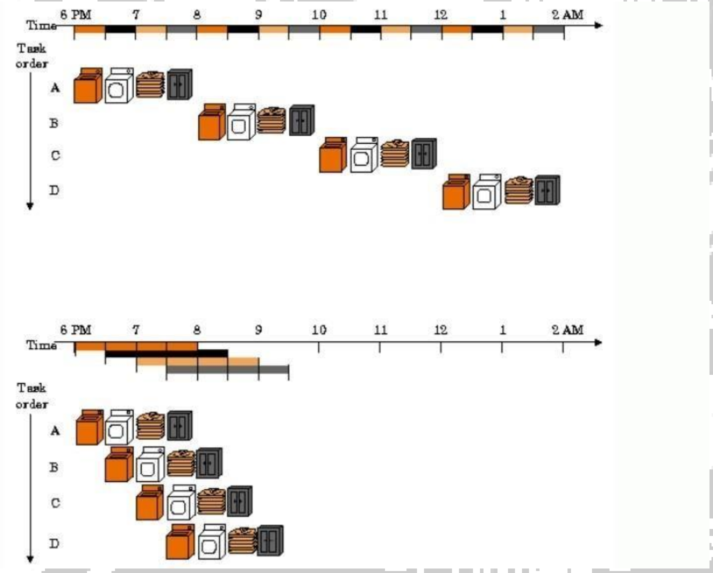

FIGURE 3.14: The laundry analogy for pipelining.

A technique used in advanced microprocessors where the microprocessorbegins executing a second instruction before the first has been completed.

A Pipeline is a series of stages, where some work is done at each stage. The work is not finished until it has passed through all stages. With pipelining, the computer architecture allows the next instructions to be fetched while the processor is performing arithmetic operations, holding them in a buffer Close to the processor until each instruction operation can perform.

<u>How Pipelines Works</u>

The pipeline is divided into segments and each segment can execute it operation concurrently with the other segments. Once a segment completes operations, it passes the result to the next segment in the pipeline and fetches the next operations from the preceding segment.

The *pipelined* approach takes much less time, as Figure 3.14 shows. As soon as the washer is finished with the first load and placed in the dryer, you load the washer with the second dirty load. When the first load is dry, you place it on the table to start folding, move the wet load to the dryer, and put the next dirty load into the washer. Next you have your roommate put the first load away, you start folding the second load, the dryer has the third load, and you put the fourth load into the washer. At this point all steps—called *stages* in pipelining—are operating concurrently.

If all the stages take about the same amount of time and there is enough work to do, then the speed-up due to pipelining is equal to the number of stages in thepipeline, in this case four: washing, drying, folding, and putting away. Therefore,pipelined laundry is potentially four times faster than non-pipelined: 20 loads would take about 5 times as long as 1 load, while 20 loads of sequential laundry takes 20 times as long as 1 load. It's only 2.3 times faster in 3.14, because we only show 4loads.

The same principles apply to processors where we pipeline instruction-execution. MIPS instructions classically take five steps:

1. Fetch instruction from memory.
2. Read registers while decoding the instruction. The regular format of MIPSinstructions allows reading and decoding to occur simultaneously.
3. Execute the operation or calculate an address.
4. Access an operand in data memory.
5. Write the result into a register.

DESIGNING INSTRUCTION SETS FOR PIPELINING

First, all MIPS instructions are the same length. This restriction makes it much easier to fetch instructions in the first pipeline stage and to decode them in the second stage.

Second, MIPS have 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 we can use the execute stage to calculate the memory address and then access memory in the following stage.

Fourth, operands must be aligned in memory. Hence, we need not worry about a single data transfer instruction requiring two data memory accesses; the requested data can be transferred between processor and memory in a single pipeline stage.