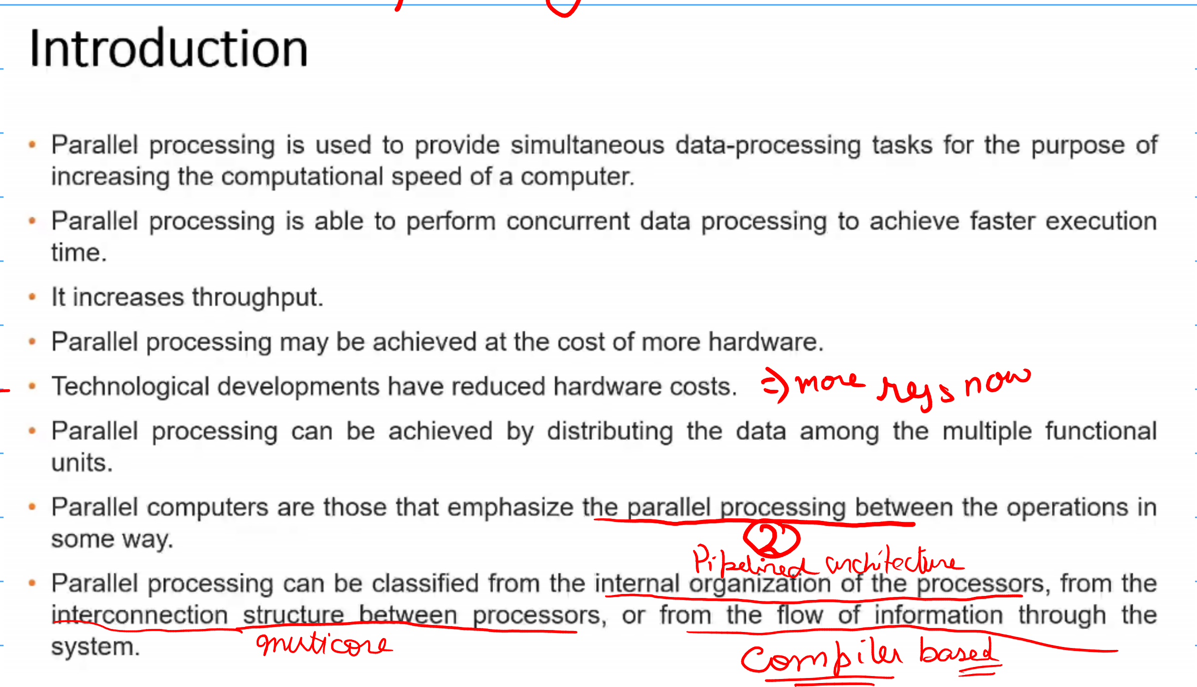
Multicycle architecture with Pipelining



Introduction

- Parallel processing is used to provide simultaneous data-processing tasks for the purpose of increasing the computational speed of a computer.
- Parallel processing is able to perform concurrent data processing to achieve faster execution time.
- It increases throughput.
- Parallel processing may be achieved at the cost of more hardware.
- Technological developments have reduced hardware costs. ⇒ more regs now
- Parallel processing can be achieved by distributing the data among the multiple functional units.
- Parallel computers are those that emphasize the parallel processing between the operations in some way.
- Parallel processing can be classified from the internal organization of the processors, from the interconnection structure between processors, or from the flow of information through the system.

② Pipelined architecture

multicore

compiler based

registers are costly modules, fastest data transmission

⟶ But we need optimal number of registers

5 functional units [3 primary]

$$\begin{tikzcd}
\text{MEM} \arrow[r] & \text{Temp} \\
\text{REG FILE} \arrow[r] & \text{registers} \\
\text{ALU} &
\end{tikzcd}$$

② Each of these steps in a multicycle architecture can be executed in parallel, provided they are executing in different functional units.

Sed

$I_1 \rightarrow S_1, S_2, S_3, S_4, S_5$ (f_1, f_2, f_3, f_4, f_5)

$I_2 \rightarrow S_1, S_2, S_3, S_4, S_5$ (f_1, f_2, f_3, f_4, f_5)

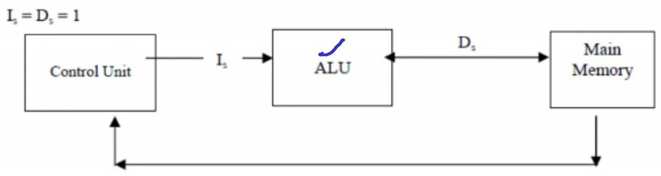
Flynn's Classification

Flynn's classification

4 categories
SISD
SIMD
MISD
MIMD

- M.J. Flynn classified computers based on the number of instructions and data that are manipulated simultaneously.
- All the computers classified by Flynn are not parallel computers.
- Instruction and data stream: refers to the flow of instruction between memory and CPU.
- Flynn's classification:
- **SISD (Single Instruction and Single Data stream):** in this organization, sequential execution of instructions is performed by one CPU containing a single processing element (PE), i.e., ALU under one control unit. (no parallelism)
- Examples of SISD: CDC 6600 which is unpipelined but has multiple functional units. CDC 7600 which has a pipelined arithmetic unit.

Only ALU



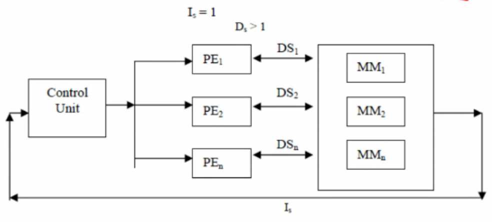
SIMD →

- **SIMD (Single Instruction and Multiple Data stream):** in this organization, multiple processing elements work under the control of a single control unit.
- All PEs receive one instruction which is operated over multiple data stream.
- Main memory can also be divided into modules for generating multiple data streams acting as a distributed memory.
- Examples of SIMD:ILLIAC-IV, PEPE, DAP.

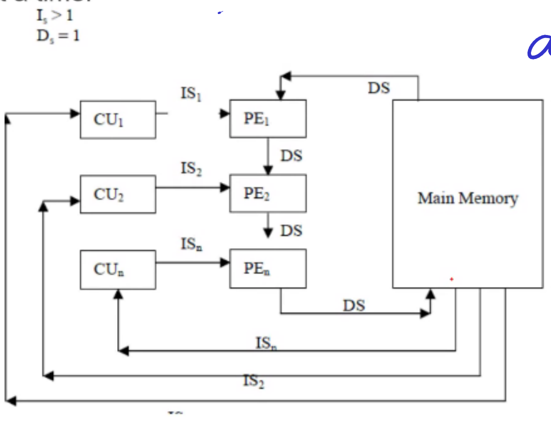
```
for (i=0; i<100; i++)
    A[i] = B[i] + c
```

eg: vector addition

requires multiple processing elements



- **MISD (Multiple Instruction and Single Data stream):** in this organization, multiple processing elements (PEs) work under the control of a multiple control units (CUs). → since each instruction is different diff clock cycles etc.
- Each control unit is handling one instruction stream and is processed through its corresponding processing element.
- Each PE is processing only one data stream at a time.
- Main memory is shared by all the CUs.
- Examples of SIMD: C.mmp developed by Carnegie-Mellon university.



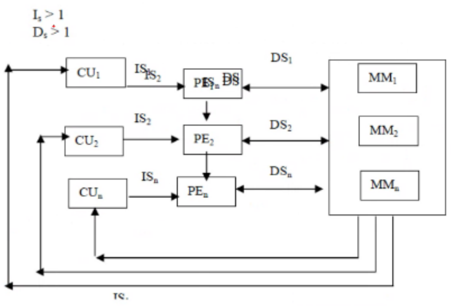
not realistic

but very useful for quantum computers

MIMD:

Multiple ins. Multiple data in the "same" time → pipelined

- **MIMD (Multiple Instruction and Multiple Data stream):** in this organization, multiple processing elements and multiple control units are organized as in MIMD. pipelined kinda
- Multiple instruction streams operate on multiple data streams.
- Multiple control units and multiple processing elements are organized such that multiple processing elements are handling multiple data streams from the main memory.
- The processors work on their own data with their own instructions.
- Tasks executed by different processors can start or finish at different times.
- Examples of MIMD: CRAY-2, IBM 370/168.



memory segmented

pipelined architecture dont necessarily have
→ Multiple Control units
→ not necessary to segment memory

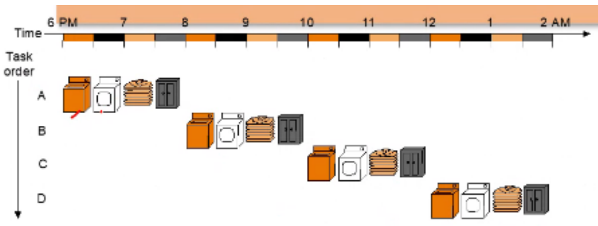
You can't normally execute more instructions by inc. clock (power leakage)

Pipeline processing

increase the throughput by reducing the (average) CPI.

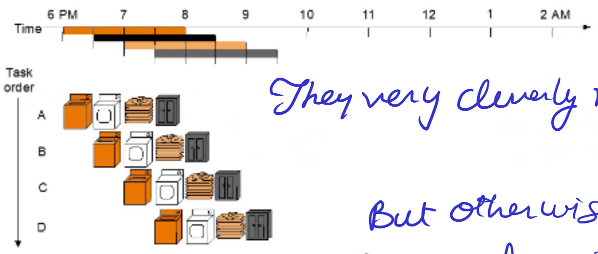
4ns

- Flynn's classification emphasizes on behavioral characteristics of the computer system rather than its operational and structural interconnections.
- Pipeline processing is a type of parallel processing. Does not fit under Flynn's classification. or MIMD
- Pipeline processing is an implementation technique where arithmetic sub-operations or the phases of a computer instruction cycle overlap in execution.
- It decomposes sequential process into sub-operations, with each sub-process being executed in a special dedicated segment that operates concurrently with all other segments.
- Each segment performs partial processing dictated by the way the task is partitioned.
- The final result is obtained after the data have passed through all the segments.
- Example: $A_i*B_i+C_i$



Not pipelined

Assume 30 min. each task – wash, dry, fold, store – and that separate tasks use separate hardware and so can be overlapped



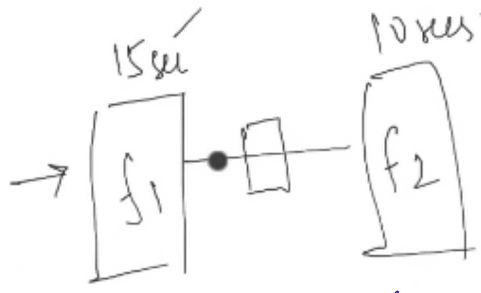
They very cleverly made every functional unit take

Pipelined same time:

But otherwise please note: ↓

Note: each functional unit takes 1 clock cycle

100 instructions



X 2500 secs (non-pipelined)

30x100 = 3000

↳ 1 clock cycle is 15 secs here.....

pipelined

⇒ 30+(15x99)

= 1515

≈ 50% improvement

Since each functional unit takes 1 clock cycle and each clock cycle must be of the largest instruction

Lets shift some of the logic of f_1 into f2

$f_1 : 13$; $f_2 : 12$ clock cycle : 13 //

Pipelining : $26 + 99 \times 13 = 1313$ > 50% improvements.

$\Rightarrow$ We should try to make each stage to take almost the same time as every other stage.