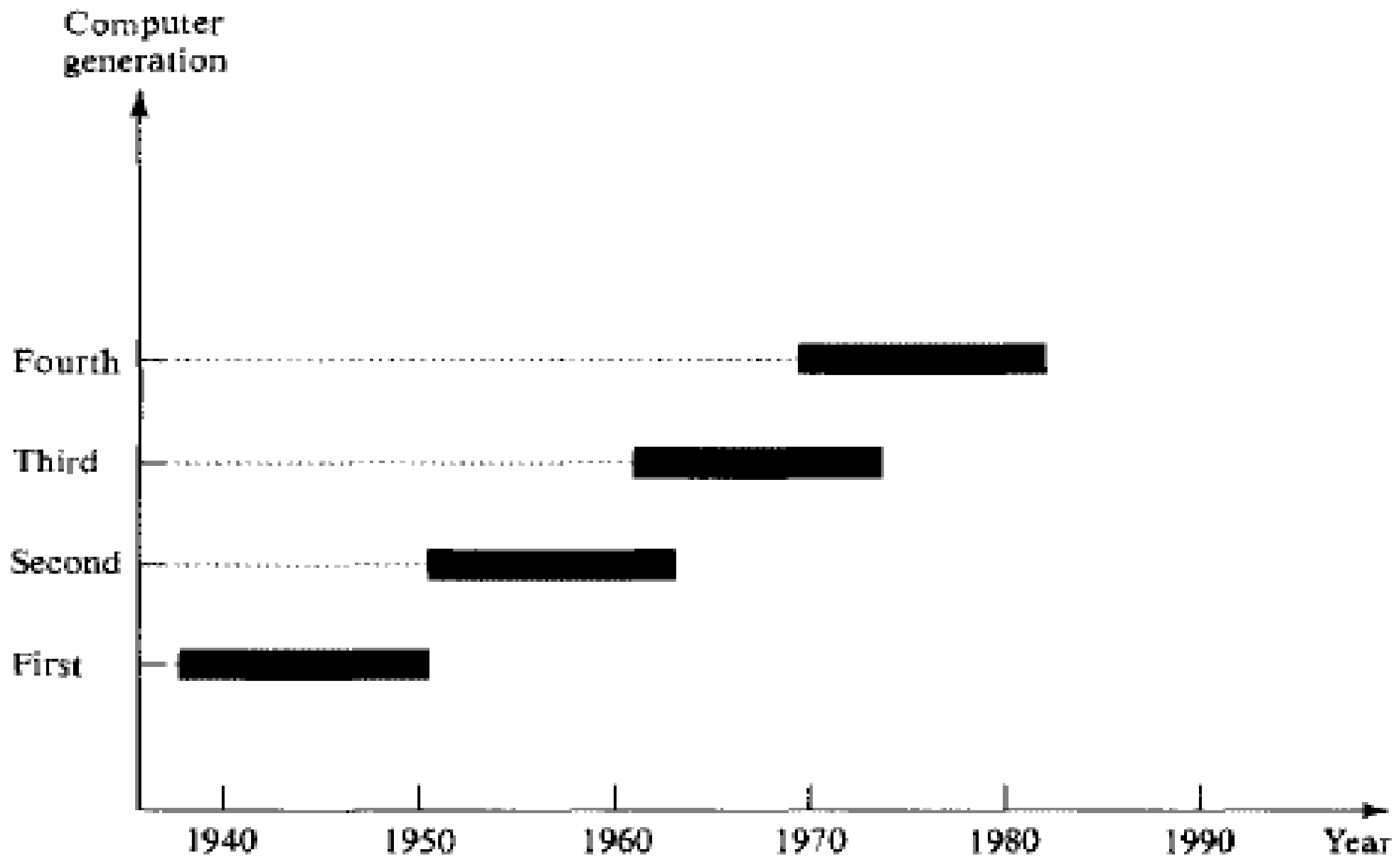
Introduction to Parallel Architectures

5 Hours

Topics covered:

❖ **Please click on the links in following slides to display the relevant figures and web pages**

Generation of Computer Systems



1. The first generation (1938-1953)

- Electromechanical relays(1940s) and Vacuum tubes(1950s) were used as switching devices
- Devices were interconnected by insulated wires
- CPU structure was bit-serial (arithmetic is done on a bit-by-bit fixed-point basis)
- Binary-coded machine language was used

2. The second generation (1952-1963)

- Transistors and Diodes were used as the building blocks
- Usage of Printed Circuit Board(PCB)
- Usage of Magnetic core memory
- Assembly language was initially used and then appeared the High-level programming languages(Fortran, Algol, Cobol)
- Batch processing was popular, providing sequential execution of user programs

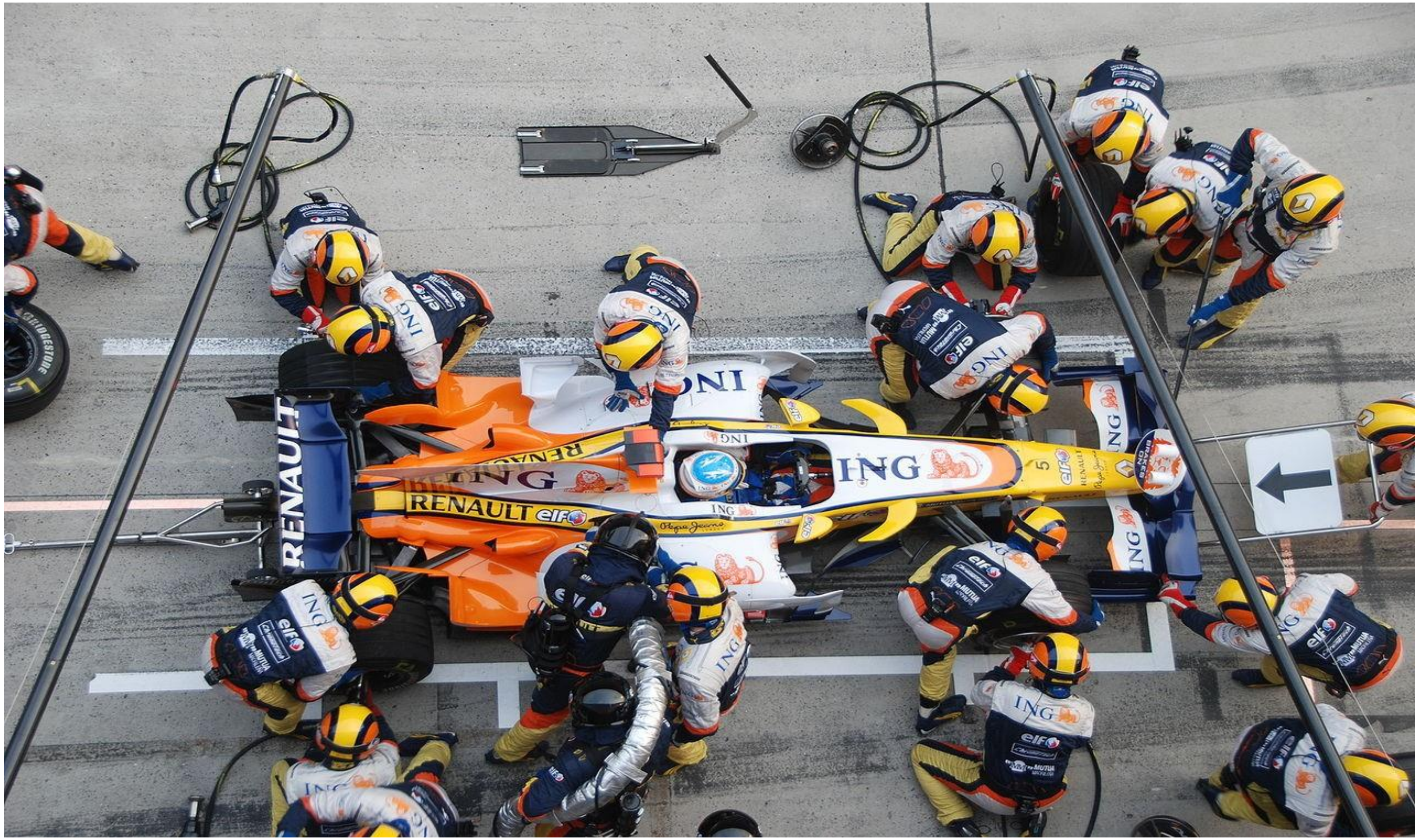
3. The third generation (1962-1975)

- Usage of Small-scale integrated (SSI) and Medium-scale integrated (MSI) circuits as the basic building blocks
- Usage of Multi-layered PCB
- Usage of intelligent compilers
- Multiprogramming is used to allow simultaneous execution of many program segments
- Time-sharing operating system became available
- Virtual memory was developed

4. The forth generation (1972-present)

- Usage of Large-scale integrated (LSI) circuits for both logic and memory sections
- High-level languages are capable of handling both Scalar and Vector data
- Pipelining and Multiprocessing become the common approaches for achieving parallelism

Parallel Programming Analogy



Realistic Expectations

- Ex. – Your program takes 20 days to run
- 95% can be parallelized
- 5% cannot (serial)
- What is the fastest this code can run?
  - As many CPU's as you want!

$\boxed{\text{1 day!}} \longrightarrow \boxed{(5/100)*20}$

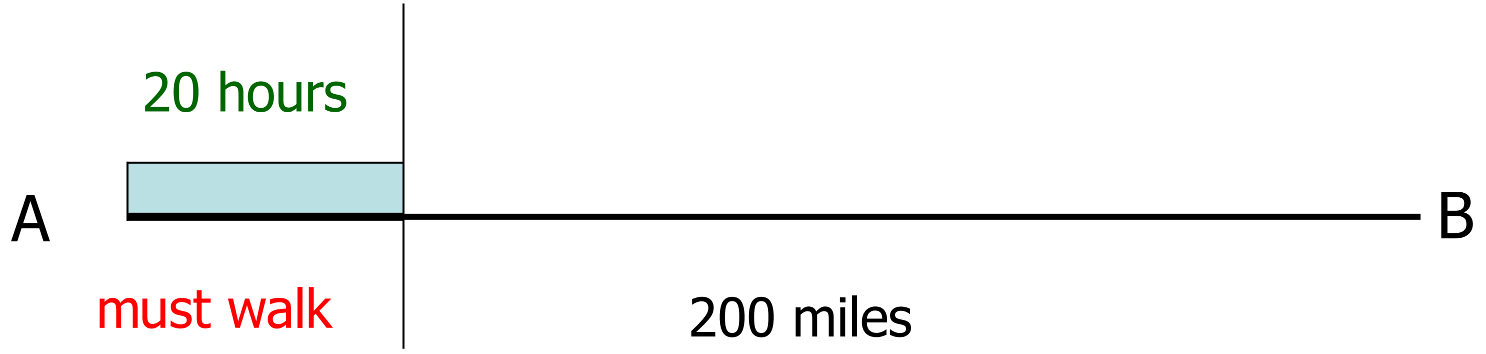
Amdahl's Law

The performance improvement to be gained from using some faster mode of execution is limited by the fraction of the time the faster mode can be used

Speedup (S)

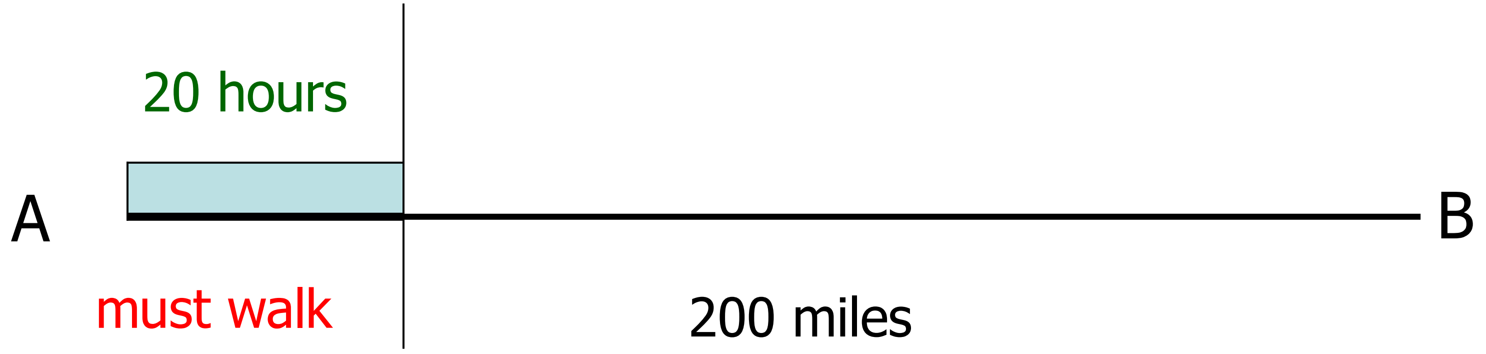
- S = Speed(new) / Speed(old)
- S = Work/time(new) / Work/time(old)
- S = time(old) / time(new)
- S = time(before improvement) /
  time(after improvement)

Example



Walk 4 miles /hour
Bike 10 miles / hour
Car-1 50 miles / hour
Car-2 120 miles / hour
Car-3 600 miles /hour

Example



Walk 4 miles /hour	→	50 + **20** = 70 hours	S = 1
Bike 10 miles / hour	→	20 + **20** = 40 hours	S = 1.8
Car-1 50 miles / hour	→	4 + **20** = 24 hours	S = 2.9
Car-2 120 miles / hour	→	1.67 + **20** = 21.67 hours	S = 3.2
Car-3 600 miles /hour	→	0.33 + **20** = 20.33 hours	S = 3.4

Introduction to Parallel Processing

The trends towards parallel processing can be seen from:

1. **Application point of view**
2. **Operating system point of view**

1. Application point of view

The computers are experiencing a trend of four ascending levels of sophistication:

- **Data Processing →** Data objects include numbers, characters, images, audio, video, multidimensional measures etc.
- **Information Processing →** Information item is a collection of data objects that are related by some syntactic structure or relation
- **Knowledge Processing →** Knowledge consists of information items plus some semantic meanings
- **Intelligence Processing →** Intelligence is derived from a collection of knowledge items.

Data-space

2. Operating system point of view

The computer systems have improved chronologically in four phases:

- Batch Processing
- Multiprogramming
- Time Sharing
- Multiprocessing

Parallel processing of information increases sharply by exploiting concurrent events

Concurrency → **implies parallelism, simultaneity, and pipelining**

Parallelism→ **Parallel events may occur in multiple resources during the *same time interval***

Simultaneity→ **Simultaneous events may occur at the *same time instant***

Pipelining→ **Pipelined events may occur in *overlapped time spans***

Degree (level) of parallel processing

Job or Program level →The highest level of parallel processing is conducted among multiple jobs or programs through multiprogramming, time sharing and multiprocessing

Task or Procedure level →The next highest level of parallel processing is conducted among procedures or tasks (program segments)

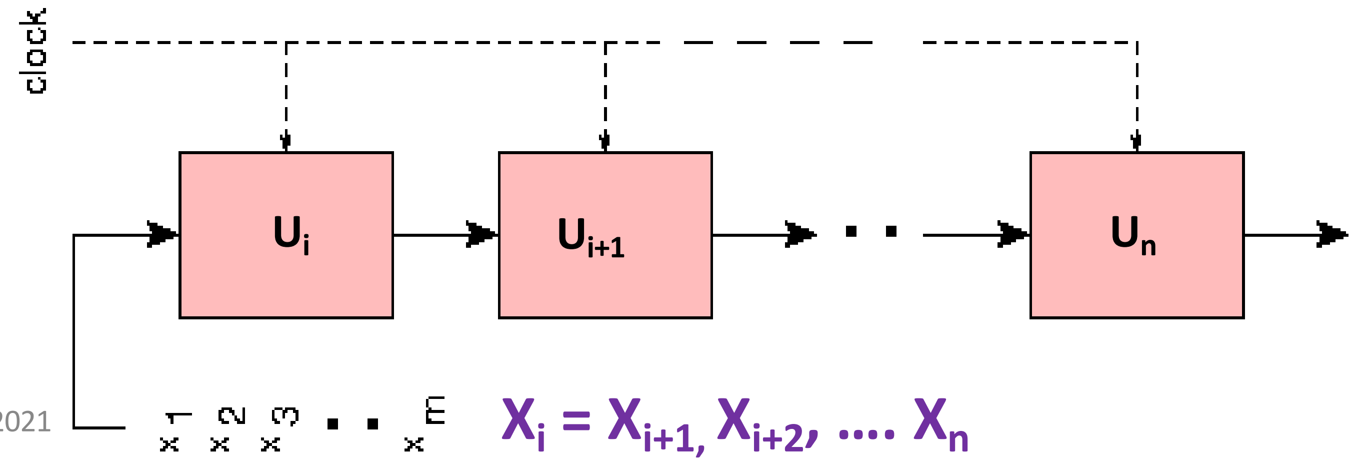
Inter-instruction level → The third level exploit concurrency among multiple instructions

Intra-instruction level → The final last level exploit concurrency within each instruction

The highest level of parallelism is often conducted algorithmically and the lowest level is implemented by hardware means

Temporal parallelism

- *Temporal parallelism* or *pipelining* refers to the execution of a task as a ***'cascade' of sub-tasks***
- There exists one functional unit to carry out each sub-task
- All these successive units can work at the same time, in an overlapped fashion
- As data are processed by a given unit $\mathbf{U_i}$, they are sent to the next unit $\mathbf{U_{i+1}}$ and the unit $\mathbf{U_i}$ restarts its processing on new data, analogously to the flow of work in a car production line. Each functional unit can be seen as a "specialized" processor in the sense that it always execute the same sub-task

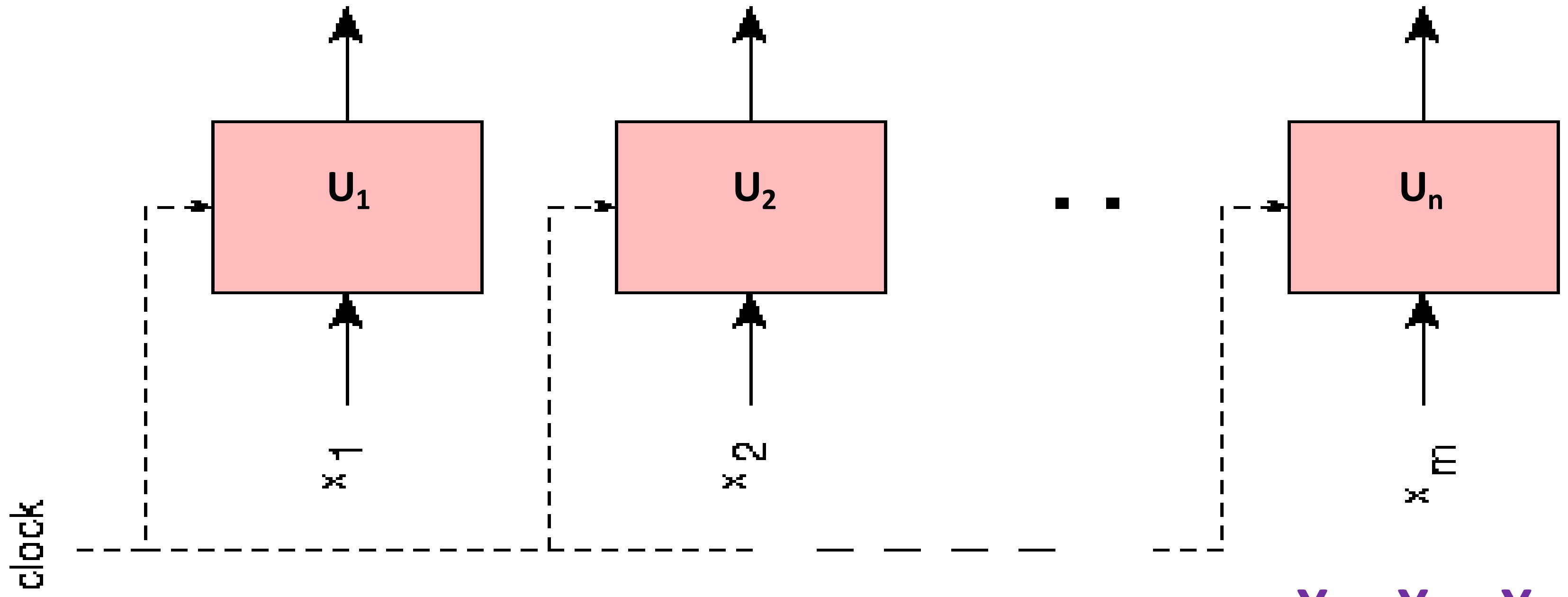


$X_i = X_{i+1}, X_{i+2}, \ldots X_n$

Spatial parallelism

- *Spatial parallelism* refers to the ***simultaneous execution*** of tasks by ***several processing units***

- At a given instant, these units can be executing the *same task* (or instruction) or *different tasks*. The former case is called ***SIMD (Single Instruction stream, Multiple Data stream)***, whereas the latter is called ***MIMD (Multiple Instruction stream, Multiple Data stream)***



$X_i = X_{i+1}, X_{i+2}, \ldots. X_n$

1. *Pipeline computers*

- Normally, the process of executing an instruction in a digital computer involves four major steps:

  1. Instruction Fetch (IF)→ fetching instruction from the main memory

  2. Instruction Decoding (ID) → identifying the operation to be performed

  3. Operand Fetch (OF) → fetching the operands is needed in the execution

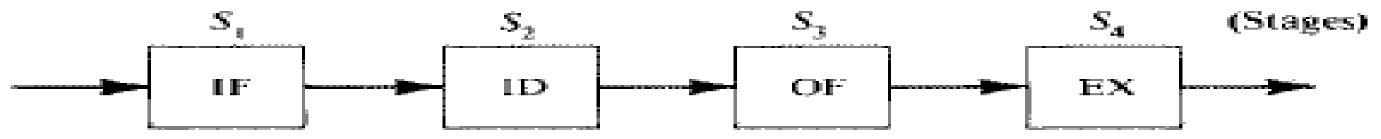
  4. Execution (EX) → Execution of the decoded arithmetic-logic operation

- In a nonpipelined computer, *these four steps must be completed before the next instruction can be issued*

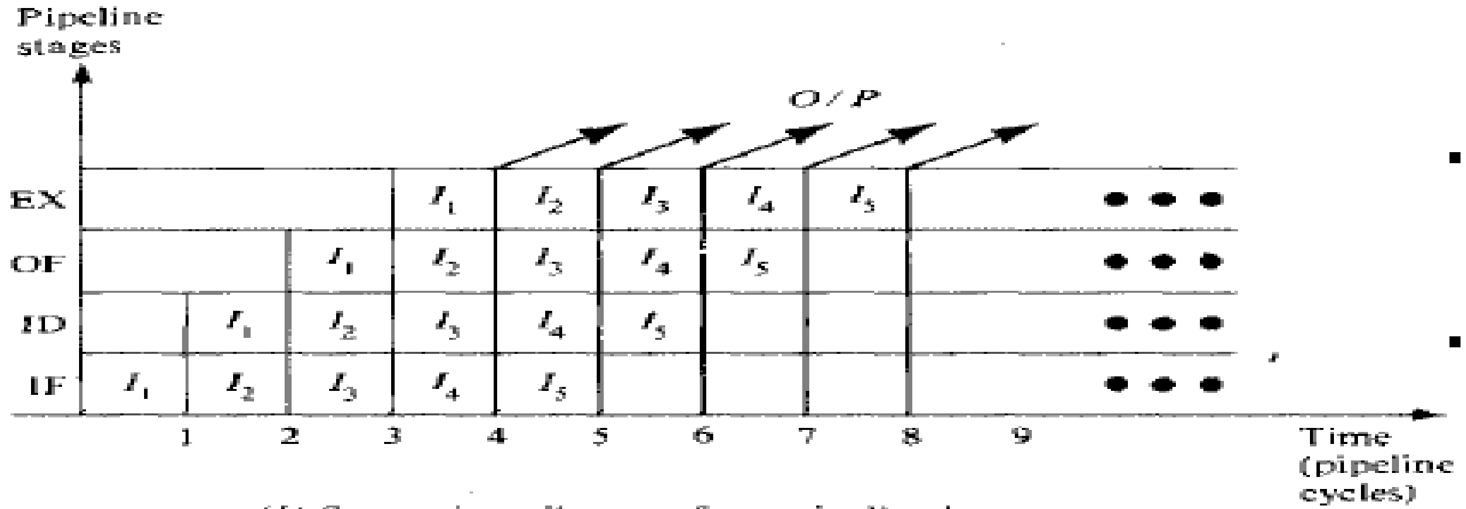
- In a pipelined computer, *successive instructions are executed in an overlapped fashion*. The Four pipeline stages, IF, ID, OF, and EX, are arranged into a linear cascade

- The two *space-time diagrams* in the next slide show the difference between overlapped instruction execution *(pipelining)* and sequentially nonoverlapped execution *(non-pipelining)*

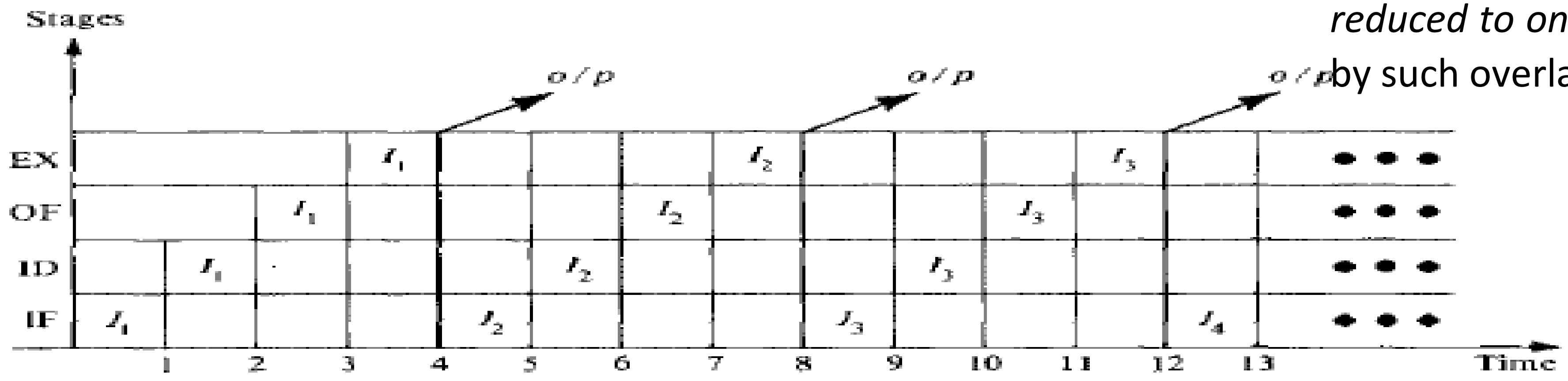
Pipeline computers



(*a*) A pipelined processor



(*b*) Space-time diagram for a pipelined processor



(*c*) Space-time diagram for a nonpipelined processor

- A pipeline cycle can be set equal to the delay of the *slowest stage*
- The flow of data (input operands, intermediate results, and output results) from stage to stage is triggered by a *common clock* of the pipeline
- The operation of all stages is *synchronized* under a common clock control
- *Interface latches* are used between adjacent segments to hold the intermediate results
- The instruction cycle has been effectively *reduced to one-fourth* of the original cycle time by such overlapped execution.

Utilising Temporal Parallelism

- 1000 candidates appear for an examination
- Answers to 4 questions in each answer book
- Teacher to evaluate answer scripts

Procedure-1

- Step1: Take an answer book from the pile of answer books
- Step2: Correct the answer to Q1, namely, A1.
- Step3: Repeat step2 for answers to Q2-Q4, namely A2-A4
- Step4: Add marks given for each answer
- Step5: Put the answer book in a pile of corrected answer books
- Step6: Repeat step 1 to 5 until all the answer books are evaluated

Time

- If a paper takes 20 minutes to correct then to correct 1000 papers 20000 minutes is required

Temporal Parallelism

- Ask four teachers to co-operatively correct each answer book.
- Four teachers sit in one line.
- The first teacher corrects answer to Q1, namely, A1 of the first paper and passes the paper to the second teacher who starts correcting A2.
- The first teacher immediately takes the second paper and corrects answer1 in it.

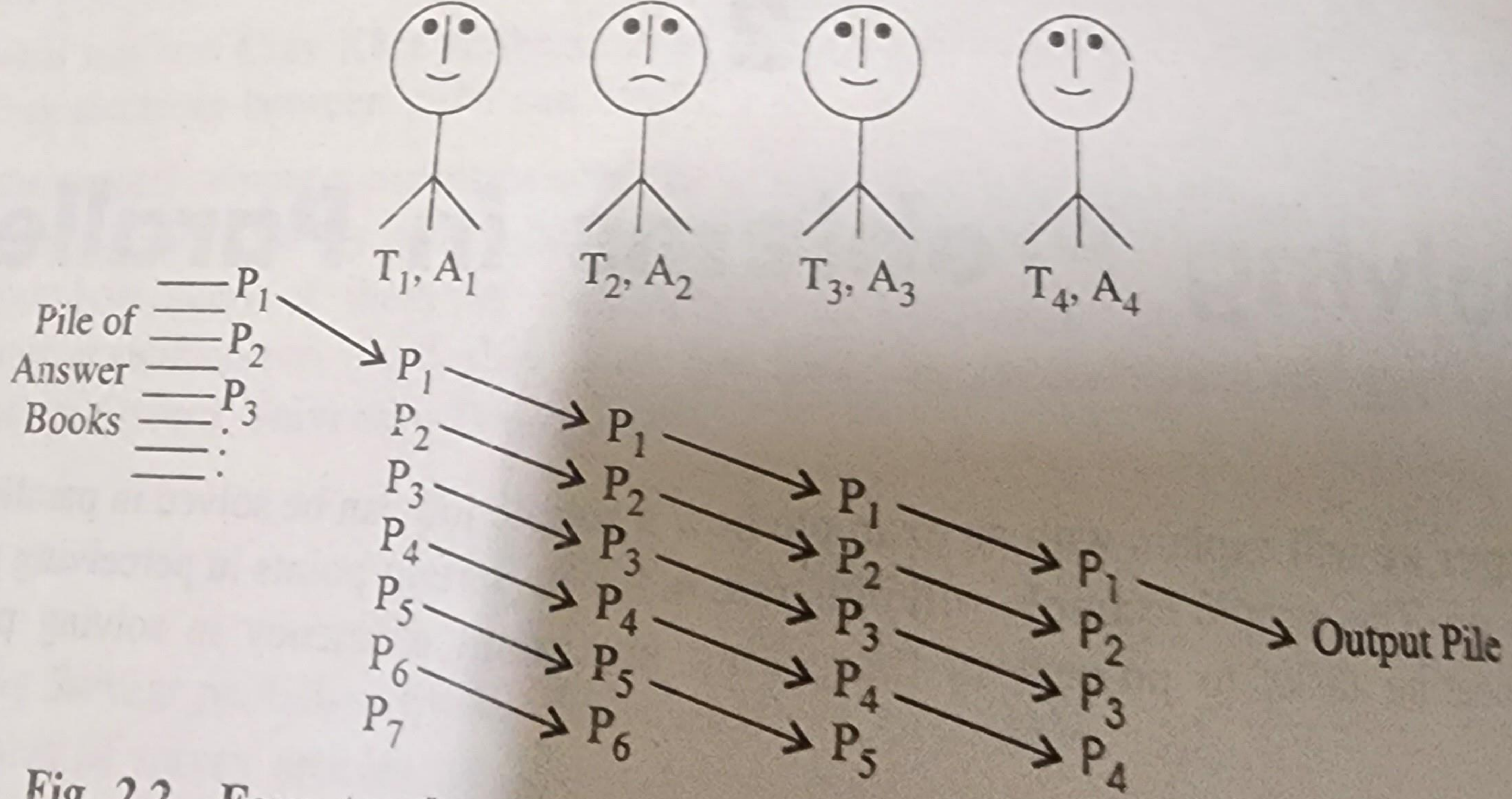


Fig. 2.2 Four teachers working in a pipeline or assembly line.

It is seen from

Temporal Parallelism

- It is seen from figure, that when the first three answer paper being corrected some teachers are idle. However from fourth paper onwards all teachers are busy correcting one answer script.
- T1-A1-P4
- T2-A2-P3
- T3-A3-P2
- T4-A4-P1

Time

- If time taken to correct each answer is 5 minutes then first answer book will take 20 minutes to correct, after that one answer book will be ready every 5 minutes.
- Time taken to correct 1000 papers will be
  - 20+(999*5)=5015 Minutes
  - Speed up is approximately 4 over sequential version

Temporal Parallelism Applicability

- 1. The jobs to be carried out are identical
- 2. A job can be divided into many independent tasks
- 3. The time taken by each task is same
- 4. The communication cost is less compared to time needed to do the task
- 5. The number of tasks is much smaller compared to the total number of jobs to be done.

Quantification

- Let
- Number of jobs=n
- Time to do a job=p
- Number of task=k
- Time for doing each task=p/k
- Time to complete n jobs with no pipeline processing=n*p

Quantification

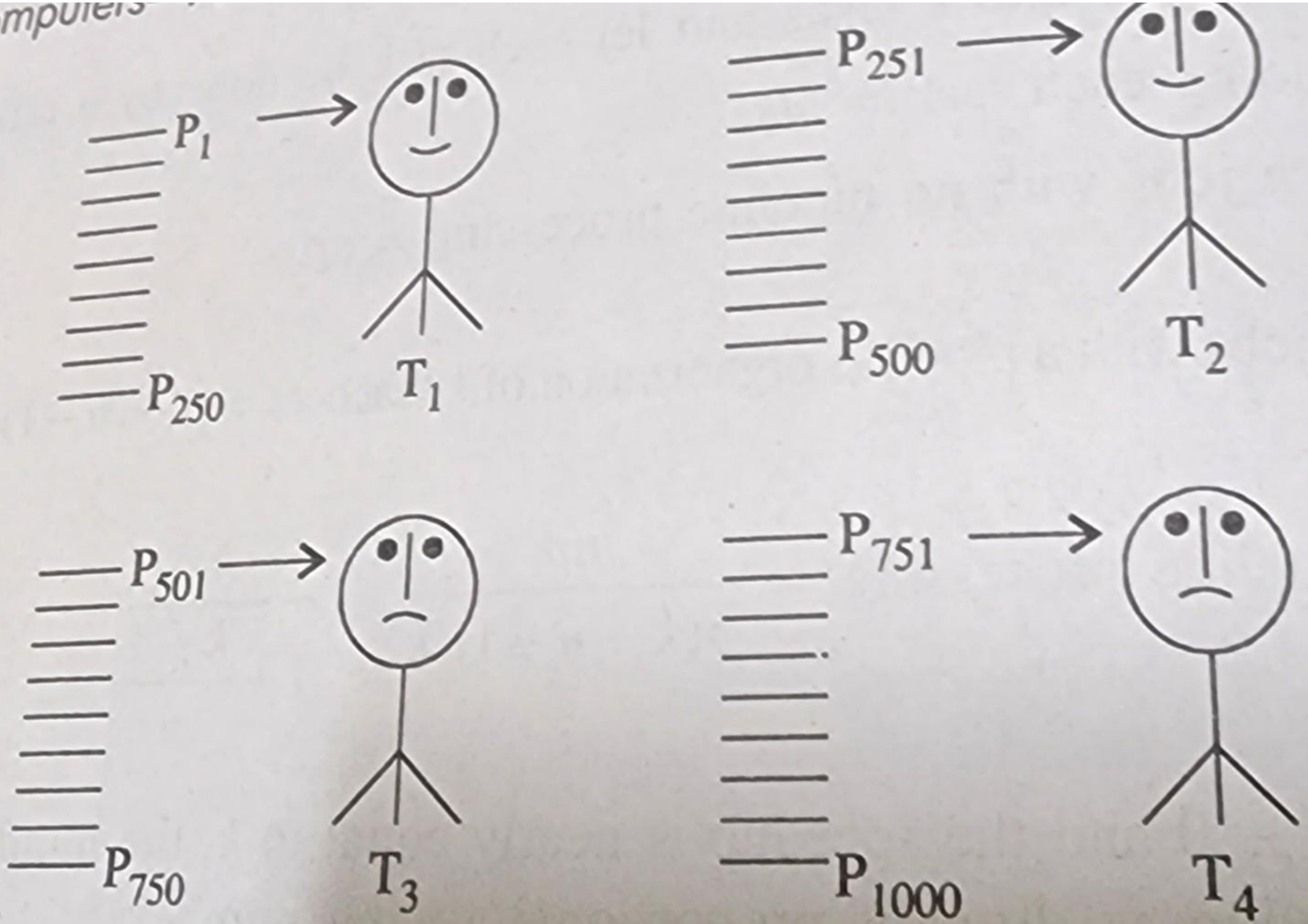
- Time to complete n jobs with a pipeline organization of k tasks
- $p+(n-1)*p/k=p*(k+n-1)/k$
- Speedup
- $(n*p)/(p*(k+n-1)/k)=k/(1+(k-1)/n)$
- If $n>>k$ then $(k-1)/n$ is approximately 0 and speedup is nearly equal to k

Problems with temporal Parallelism

- **Synchronisation**: Identical time should be taken for doing each task in the pipeline so that a job can flow smoothly in the pipeline without holdup
- **Bubbles in pipeline**: If some tasks are absent in a job bubbles form in the pipeline.
- **Fault tolerance**: The system does not tolerate faults.
- **Inter-task communication**: The communication cost between tasks should be as low as possible
- **Scalability**: The number of stages in the pipeline cannot be increased beyond a certain limit.

Data Parallelism

- We divide the answer books into four piles and give one file to each teacher. Each teacher follows sequential instructions(refer to sequential algorithm)
- Assuming each teacher takes 20 minutes to correct an answer book, the time taken to evaluate 1000 answer books is 5000 minutes as each teacher corrects only 250 papers and all teachers work simultaneously.


P_1
P_{250}
T_1
P_{251}
P_{500}
T_2
P_{501}
P_{750}
T_3
P_{751}
P_{1000}
T_4

Quantification

- Number of jobs=n
- Time required to do a job p
- Let there be k teachers to do the job
- Let the time required to distribute the jobs to k teachers be k*q
- Time to complete n jobs by k teachers= k*q+(n*p)/k
-

Speedup

- $(n*p)/((k*q)+(n*p)/k)=k*n*p/((k^2)*q+n*p)=k/(1+(k^2)*q/(n*p)$
- If $(k^2)*q<<n*p$ then speedup is nearly equal to k.

Efficiency

- It is the ratio of actual speedup to the maximum possible speedup

Problems

- 1. n=1000 p=20 k=4 and q=1 find Speedup and efficiency
- 2. n=1000 p=20 k=100 and q=1 find speedup and efficiency

Solution

- 1. Speedup=3.997 and Efficiency=0.99
- 2. Speedup=66.7 and Efficiency=0.667

Data Parallelism Advantages

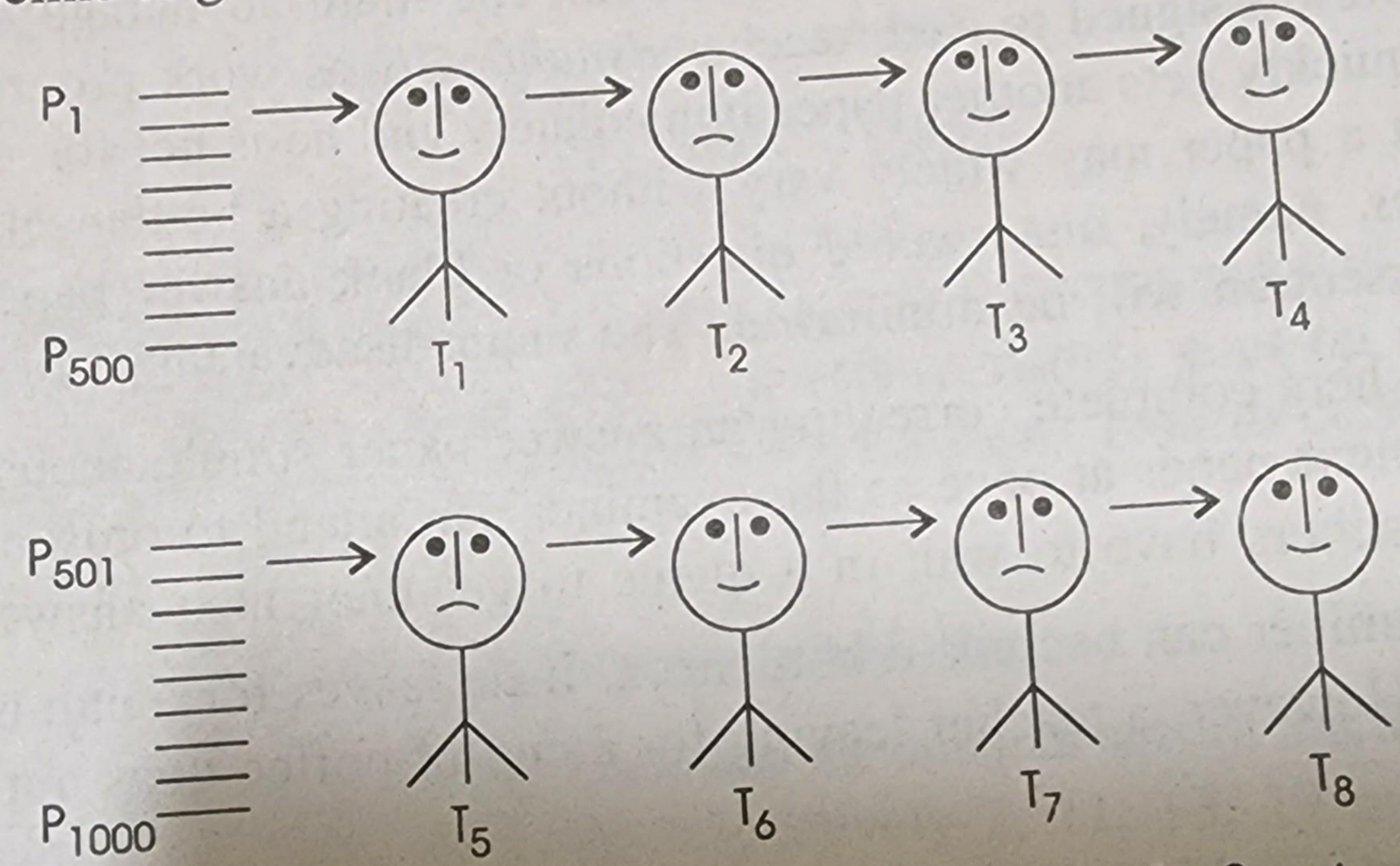
- There is no synchronisation required
- The problem of bubble is absent
- More fault tolerant
- Intertask communication not required

Data Parallelism Disadvantages

- Static assignment of jobs is inefficient
- The set of jobs must be partitionable into subsets of mutually independent jobs. Each subset should take the same time to complete
- All the instructions needs to be executed by each process.
- The time taken to divide a set of jobs into equal subsets of jobs should be small. Further, the number of subsets should be small compared to the number of jobs

method 1 and method

pipeline is given half the total number of jobs. This is

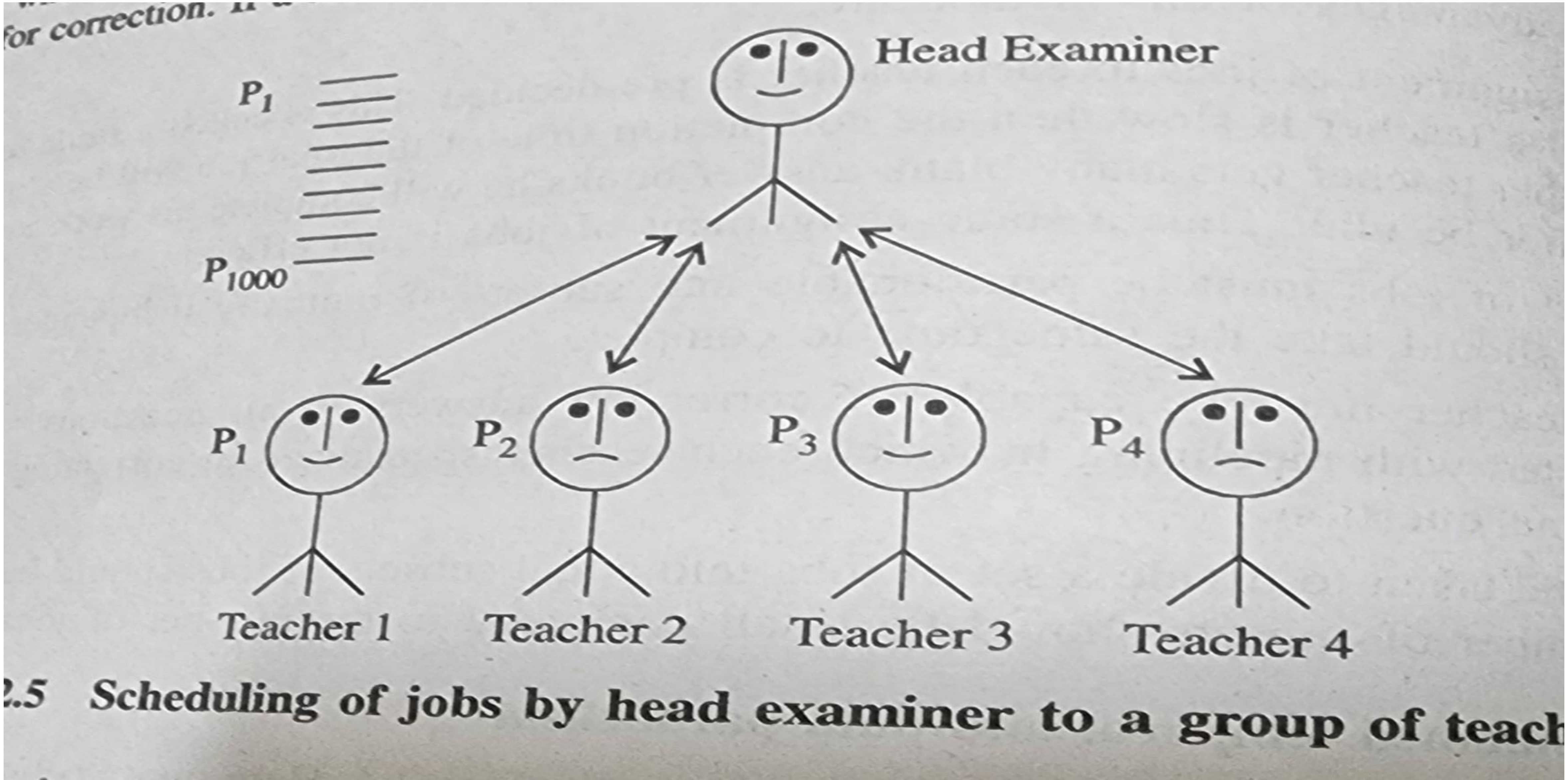


ght teachers working in two pipelines to grade 2 sets of answ

Combined temporal and data parallelism

- It is also known as parallel pipeline processing
- This method almost halves the time taken by a single pipeline
- If it takes 5 minutes to correct an answer, the time taken by two pipelines is (20 + 499 * 5)=2515 minutes

Data parallelism with dynamic assignment


for correction. [illegible]

.5 **Scheduling of jobs by head examiner to a group of teach**

Data parallelism with dynamic assignment

1. Head examiner gives one answer book to each examiner and keeps the rest of answer book with him.
2. All teachers simultaneously correct the paper given to them.
3. A teacher who completes early will go back to the head examiner to collect another answer book.
4. If a second teacher completes at the same time , then he queues up in front of head examiner and waits for his turn to get an answer paper.
5. The procedure is repeated till all the answer papers are corrected.

Advantages

1. Load balancing of work
2. The process is not affected by bubbles.
3. The overall time taken for paper correction will be minimized.

Disadvantages

1. If many teachers correct answer book simultaneously then they have to wait in a queue to get answer book
2. The head examiner can become a bottleneck.
3. The head examiner is idle between handing out papers.
4. Not scalable

Quantification

Let total number of papers=n
Average time to correct a paper = p
Sequential = n*p
Number of teachers = k
Let the time a teacher waits for getting and returning an answer book = q
Time taken by each teacher to get, evaluate, and return an answer book= p+q
Assuming that each teacher corrects (n/k) papers and all teachers work simultaneously
Total time taken to correct n scripts by k teachers= n*(p+q)/k

Speedup

Speedup=n*p/(n*(p+q)/k)

=k/(1+q/p)

As long as q<<p the speedup approaches ideal value

Usually q is a function of k and goes up with k thus reducing speedup.

If q=mk then

Speedup=k/(1+m*k/p)

=1/((1/k)+(m/p))

When k is large speedup=p/m a constant

Thus procedure is not scalable unless (m*k/p)<<1

Data Parallelism with quasi-dynamic scheduling

1. Dynamic scheduling can be made still better by giving each teacher unequal sets of answer papers to correct. Once they complete their work they may be given small bunches of papers.
2. This will randomize the job completion time of each teacher and reduce the probability of queue formation before head examiner.

Comparison of temporal and data parallel processing

1. Job is divided into a set of independent tasks and tasks are assigned for processing	1. Full jobs are assigned for processing
2. Tasks should take equal time. Pipeline stages thus be synchronized	2. Jobs may take different times. No need to synchronize the beginning of the jobs
3. Bubbles in job leads to idling of processors	3. Bubbles do not cause idling of processors
4. Processors specialized to do specific tasks efficiently	4. Processors are general purpose
5. Task assignment static	5. Job assignment may be static, dynamic and quasi dynamic
6. Not fault tolerant	6. Tolerates processor faults
7. Efficient with fine grained tasks	7. Efficient with coarse grained tasks and quasi dynamic scheduling
8. Scales well as long as number of jobs is much greater than number of tasks and communication cost from one processor to next is negligible	8. Scales well as long as number of jobs is much greater than number of processors and processing time is much higher than the time to distribute data to processors