UNIT-5

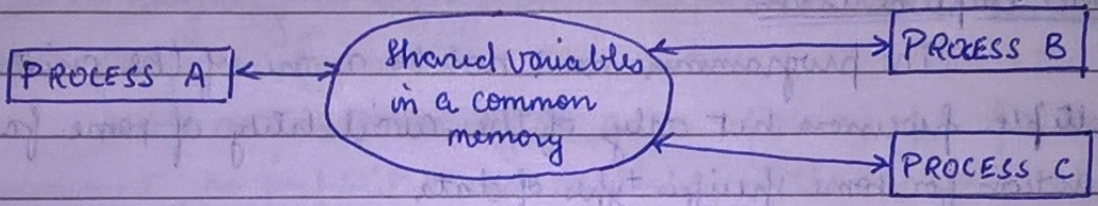
PARALLEL PROGRAMMING MODELS

① A model for parallel programming is an abstraction and is machine architecture independent. Two categories -

(1) Implicit parallelism - Compiler and runtime support system automatically specify parallelism.

(2) Explicit parallelism - Specified by the programmer in source code using special language constructs, compiler directives or library calls.

② Shared variable model -



(IPC) Interprocess communication using shared variable

→ Critical Section (CS) - It is a code of segment accessing shared variables, which must be executed by only one process at a time and which once started must be completed without interrupt. ←

CS satisfies the following requirements -

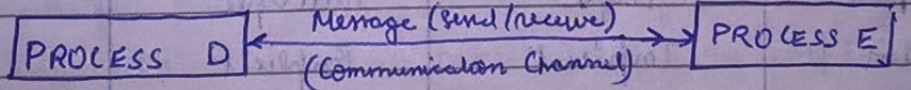
(1) Mutual exclusion

(2) No deadlock in waiting

(3) No preemption - No interrupt until completion

(4) Eventual Entry

③ Message passing model -



IPC using message passing

Message can be instructions, data, signals, interrupt etc.

Synchronous message passing - A synchronous communication is not complete until the message has been received.

Asynchronous message passing - An asynchronous communication completes as soon as the message is on the way. Sends
↳ Sends queues of message
Asynchronous message passing allows more parallelism than synchronous message passing

③ Data Parallel Model -
It originates from vector programming where the programmer writes his applications in terms of highly optimized vector operations
SIMD implementation -
The programmer need not be aware of the existence of multiple processors but only of the availability of some fast built-in operations for some specific types of data.
MIMD implementation -
The processes are only synchronized at the beginning and end of parallel operations as opposed to matching through the same sequence of instructions all at the same pace.
→ Array language extension - It is represented by high level data types. The array syntax enables the removal of some nested loops in the code and should reflect the architecture of the array processor

④ Comparison between explicit parallel programming models -

Main Features	Data Parallel	Message Passing	Shared Variable
Control Flow	Single	Multiple	Multiple
Synchrony	Loosely synchronous	Asynchronous	Asynchronous
Address Space	Single	multiple	multiple
Interaction	Implicit	Explicit	Explicit
Data allocation	Implicit or semiexplicit	Explicit	Implicit or semiexplicit

⑤ Object-oriented model -

It provides suitable abstractions and software engineering ~~models~~ methods for structured application design.

→ Concurrent Object-oriented programming (COOP) -

It combines concurrency and object-oriented programming
Eg:- JAVA, combine OOPP with threads.

Systems where objects themselves are a concurrency primitive, such as when objects are combined with the actor model.

→ Actor model -

Everything in the system is taken to be an actor. Actors are completely independent on other actors.

All actions taken by an actor upon receipt of a message are concurrent, no implicit serial ordering of the actions in a method.

Actor primitives are Create, Send-to and Become

→ Parallelism in COOP -

(1) Pipeline concurrency
(2) Divide and Conquer concurrency.
(3) Co-operative problem solving -

⑥ Functional and logic models -

→ Functional programming models -

Functional programming languages use declarative semantics and some form of lambda ~~activities~~ calculus to express the operation of a program.

Concurrency is done with pure functional languages

→ Logic programming models -

Logic programming is suitable for knowledge processing dealing with large databases. This model adopts an implicit search strategy and support parallelism in the logic inference process.

Concurrency (Three ways) and-parallelism (execute multiple predicates), or-parallelism (execute multiple guards), through explicit mapping of predicates linked together through logic assignment variables.

⑦ Parallel languages and Compilers -

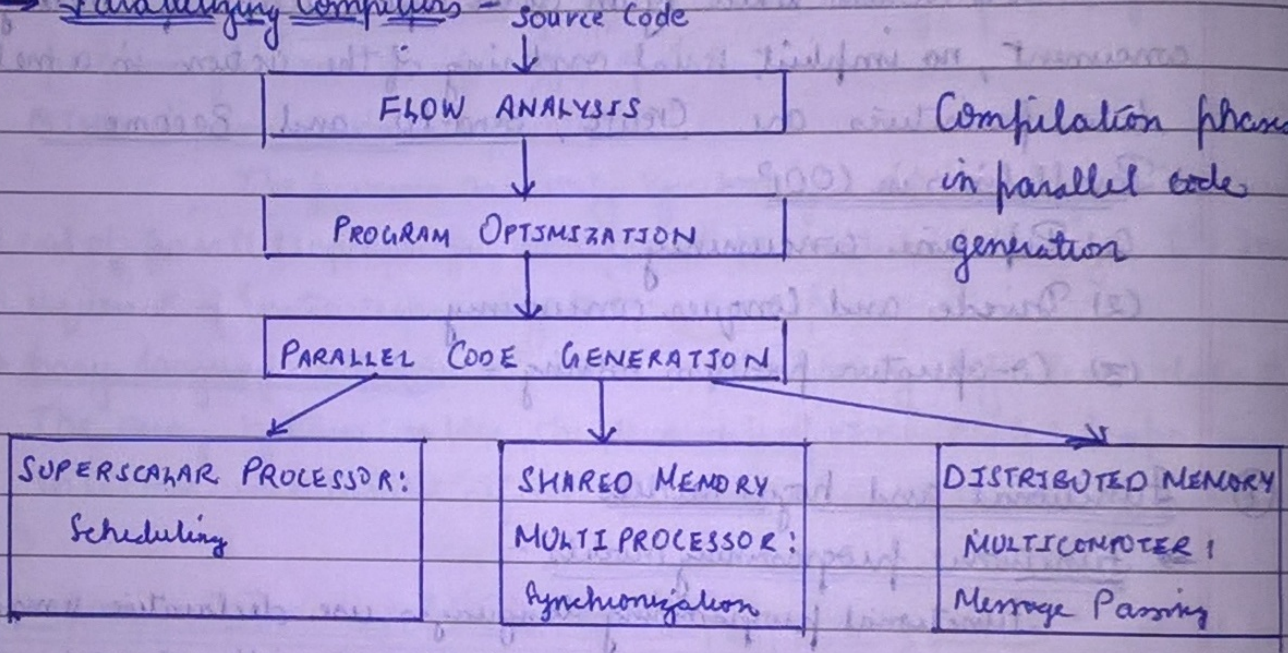
→ Parallel languages -

languages that has parallel programming Eg - SISAL, PCN

language features for parallelism are -

(1) Optimization features - (Automated analyzer, semiautomated analyzer)
(2) Availability features (Scalability, compatibility and portability)
(3) Synchronization / Communication features (Send/receive, remote procedure call)
(4) Control of parallelism (grain, explicit/implicit parallelism)
(5) Data parallelism features (Mapping functions, virtual processor support)
(6) Process Management features (lightweight processes, automatic load balancing, dynamic process creation)

→ Parallelizing Compilers -



Compilation phases in parallel code generation

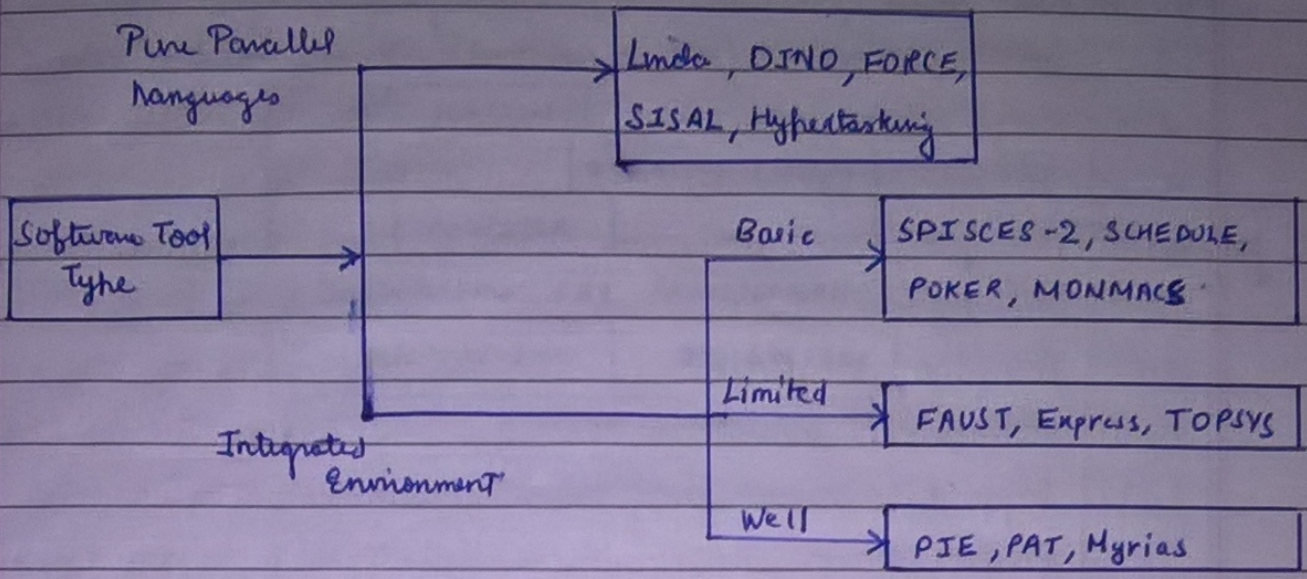
⑧ Parallel programming Environment -

It is a bridge between a system developer's natural model of an application and an implementation of that application on available hardware.

It consists of hardware platforms, language supported, OS and software tools and application packages.

It must offer portable, efficient and easy-to-use tools for development of applications.

⑨ Software tools and environments -



Basic environment provides a simple program tracing facility for debugging and performance ~~from~~ monitoring

limited integration provides tools for parallel debugging, performance monitoring or program visualization

Well developed environment provide intensive tools for debugging programs, interaction of textual/graphical representations of a parallel program, visualization support for performance monitoring, parallel I/O, parallel graphics etc.

→ Environment features -

(1) Control flow graph generation
(2) Integrated graphical maps
(3) Parallel debugger at source code level
(4) Performance monitoring by either software or hardware means
(5) Performance prediction model
(6) Parallel input/output for fast data movement
(7) OS support for parallelism in front end and back end environment