Estimating Program Run Time

Estimating Program Run Time

- A real-time program is defined as a program for which the correctness of operation depends on the logical results of the computation and the time at which the results are produced.
- In general, there are three types of programming : Sequential, multi-tasking and real time.
- Estimating program run time depends on following factors :
  1. Source code : Source code that is carefully tuned and optimized takes less time to execute.
  2. Compiler : It maps source level code into a machine level program.
  3. Machine architecture : Executing program may require much interaction between the processor and the memory and I/O devices.
  4. Operating system : OS determines such isssues as task scheduling and memory management. Both have major impact on memory management.

Analysis of Source Code

- Consider the following code :

```
a := b x c;
b - d + e;
d = e - f;
```

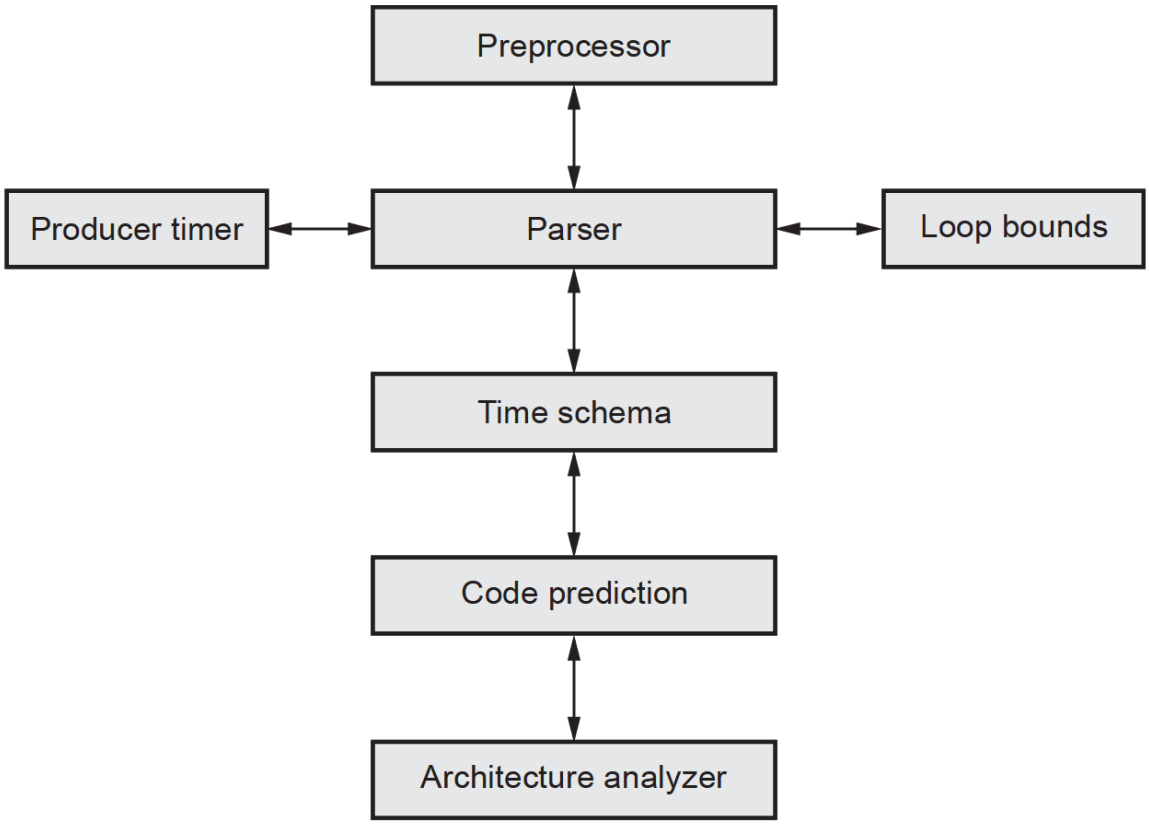
- This is straight line code. The total execution time is given by,

$$\sum_{i=1}^{3} T_{exec}(Li)$$

Where Texec(Li) is the time needed to execute Li.

- Execution time analysis is any structured method or tool applied to the problem of obtaining information about the execution time of a program or parts of a program.

- The fundamental problem that a timing analysis has to deal with is the following : The execution time of a typical program (or other relevant piece of code) is not a fixed constant, but rather varies with different probability of occurrence across a range of times.

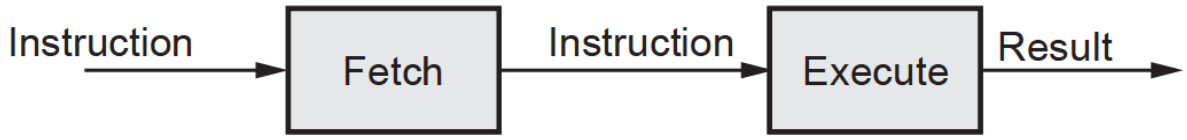


Schematic of timing estimation system

- Preprocessor produces compiled assembly language code. Parser analyze input source program.
- Procedure timer maintains a table of procedures and their execution times.
- The loop bounds module obtaines bounds on the number of iterations for the various loops in the system.
- The time schema is independent of the system, it depends only on the lanugaue.
- Code prediction module does this by using the code generated by the preprocessor and using the architecure analyzer to include the influence of the architecture.

Accounting for Pipeling

- Pipelining is the ability to overlap execution of different instructions at the same time.
- In a 2 - stage pipeline, you break down a task into two sub-tasks and execute them in pipeline. Lets say each stage takes 1 cycle to complete.
- That means in a 2-stage pipeline, each task will take 2 cycles to complete (known as latency).
- An instruction has a number of stages. The various stages can be worked on simultanously through various blocks of production. This is a pipeline. This process is also referred as instruction pipeling.
- The pipeline of two independent stages : Fetch instruction and execusion instruction. The first stage fetches an instruction and buffers it.



Pipeline of two independent stages

- While the second stage is executing the instruction, the first stage takes advantage of any unused memory cycles to fetch and buffer the next instruction. This process will speed up instruction execution.
- Several factors serve to limit the pipeline performance. If the six stage are not of equal duration, there will be some waiting involved at various pipeline stage.
- Another difficulty is the condition branch instruction or the unpredictable event is an interrupt. Other problem arise that the memory conflicts could occur. So the system must contain logic to account for the type of conflict.