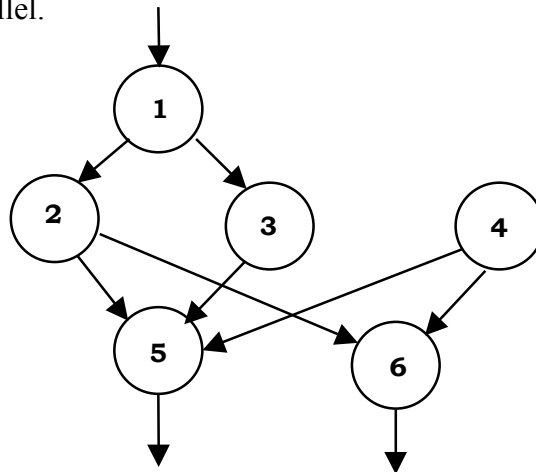


Simple Example of PRAM with EREW**Problem:**

Given a shared memory four processor system with 6 Memory Modules, and six tasks (instructions) to be executed with different data. The Memory access requirements and data dependency graph are given below. The processors can be configured as MIMD or SIMD, and for SIMD, only the **same instruction** can access different memory modules simultaneously. For MIMD there is no such restriction. In either case, the memory model is UMA, EREW, and the lower numbered processor gets priority in a memory conflict. Assume that each sub-task by an individual processor takes 1 clock cycle, which includes the required memory access. What are the performance parameters in SIMD and MIMD modes?

		Memory Access Requirements			
		Processors			
Instructions		P ₁	P ₂	P ₃	P ₄
I ₁		M ₂	M ₄	M ₃	M ₂
I ₂		M ₁	M ₃	M ₂	M ₆
I ₃		M ₄	M ₆	M ₅	M ₆
I ₄		M ₃	M ₄	M ₄	M ₂
I ₅		M ₂	M ₂	M ₂	M ₁
I ₆		M ₁	M ₅	M ₅	M ₁

Data Dependency graph for the 6 tasks, each of which can be shared equally by 4 processors operating in parallel.

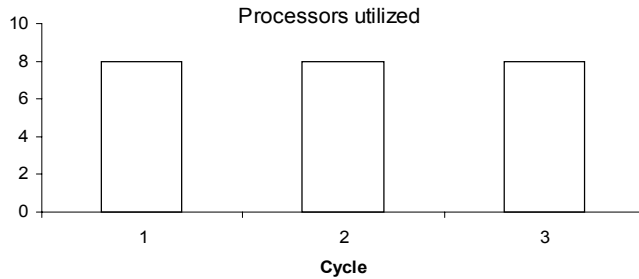


Additional Constraint: Each instruction cannot be spread over more than 3 consecutive memory cycles.

Solution:

There are 24 execute-memory operations for the entire computation. Hence the time required on an SISD uniprocessor, running at the same rate as the multiprocessors, is $T_1 = 24$ cycles. This is sometimes referred to as the "sequential time." The total workload, $W = 24$ instructions, and there are 24 memory accesses.

The parallelism profile for the algorithm is determined by the precedence relationship, which describes the dependencies of the 6 coarse-grained tasks. Since we are given no other information about the data dependencies of the fine-grained tasks, we must assume that each of the major tasks must be completed before starting any other major task that depends on it. We can see that tasks 1 and 4 are independent and can be done in parallel. Tasks 2 and 3 depend on task 1, but can be done in parallel with one another. Similarly tasks 5 and 6 can be done in parallel, as soon as tasks 2, 3, and 4 are completed. Since each task involves 4 instructions that may be done in parallel, we have the following theoretically possible profile:



During cycle 1, tasks 1 and 4 are being done. During cycle 2, tasks 2 and 3; and during the last cycle tasks 5 and 6.

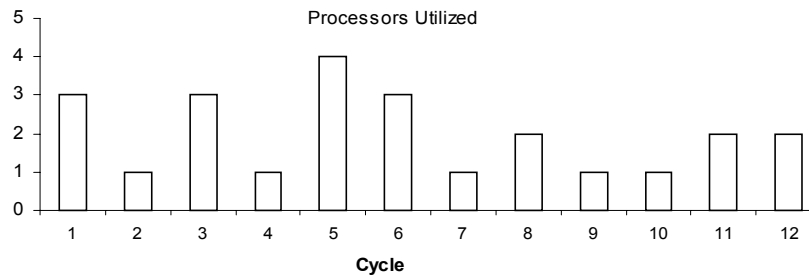
The algorithm's critical parallel path time, T_{∞} can be seen from the above profile by adding the time to do each of the 3 pieces of the workload with the degree of parallelism possible for each piece. Thus, trivially for this problem, $T_{\infty} = 8/8 + 8/8 + 8/8 = 3$.

We now construct a space-time diagram for each configuration, using our fundamental unit cycle as the time unit, tracing through the events cycle by cycle, showing the instruction being executed and the memory access being made by each processor at each cycle.

SIMD Operation:

CYCLE	P ₁	P ₂	P ₃	P ₄
1	I ₁ M ₂	I ₁ M ₄	I ₁ M ₃	----
2	----	----	----	I ₁ M ₂
3	I ₄ M ₃	I ₄ M ₄	----	I ₄ M ₂
4	----	----	I ₄ M ₄	----
5	I ₂ M ₁	I ₂ M ₃	I ₂ M ₂	I ₂ M ₆
6	I ₃ M ₄	I ₃ M ₆	I ₃ M ₅	----
7	----	----	----	I ₃ M ₆
8	I ₅ M ₂	----	----	I ₅ M ₁
9	----	I ₅ M ₂	----	----
10	----	----	I ₅ M ₂	----
11	I ₆ M ₁	I ₆ M ₅	----	----
12	----	----	I ₆ M ₅	I ₆ M ₁

The actual parallelism profile realized on the SIMD shared memory machine looks quite different from the theoretical profile:



The difference between the algorithm parallelism and the actual machine is due to load imbalance caused by memory conflicts, and also by the fact that the maximum degree of parallelism in the machine is only 4, when in fact 8 could be used by the computation.

We see that the time required for the 4 processor system, $T_n = 12$ cycles. There are a total of 24 memory references required to execute all 6 instructions for all the data. This is done in 12 total cycles. Hence, the average memory bandwidth delivered is $B_M = 24/12 = 2$ words/cycle. The speedup, $S_n = T_1/T_n = 24\text{cycles}/12\text{cycles} = 2$. The efficiency, $E_n = S_n/n = 2/4 = 0.5$.

The space-time diagram should not be confused with the **workload distribution**, which shows the portions of the total workload that can be done in parallel. From the above we can see that:

$W_1 = 5$ (instructions that can be done by only 1 processor)

$W_2 = 6$ (instructions that can be worked on by 2 processors at the same time)

$W_3 = 9$, and $W_4 = 4$.

We can see that the total workload $W = W_1 + W_2 + W_3 + W_4 = 24$ instructions, as expected. If we knew the workload distribution a priori, we could compute the n-processor execution time without the actual space-time diagram:

$$T_n = \sum_i \frac{W_i}{i} = \frac{5}{1} + \frac{6}{2} + \frac{9}{3} + \frac{4}{4} = 12 \text{ cycles, as before.}$$

MIMD Operation:

CYCLE	P ₁	P ₂	P ₃	P ₄
1	I ₁ M ₂	I ₁ M ₄	I ₁ M ₃	----
2	I ₄ M ₃	I ₄ M ₄	----	I ₁ M ₂
3	I ₂ M ₁	I ₃ M ₆	I ₄ M ₄	I ₄ M ₂
4	I ₃ M ₄	I ₂ M ₃	I ₂ M ₂	I ₂ M ₆
5	I ₆ M ₁	I ₅ M ₂	I ₃ M ₅	I ₃ M ₆
6	I ₅ M ₂	----	I ₆ M ₅	I ₅ M ₁
7	----	I ₆ M ₅	I ₅ M ₂	I ₆ M ₁

Hence $B_M = 24/7 = 3.43$ words/cycle, $T_n = 7$ cycles, $S_n = 24/7 = 3.43$, and $E_n = 3.43/4 = 0.86$. The MIMD workload distribution is different than a SIMD. For the MIMD:

$$W_1 = 0, W_2 = 0, W_3 = 12, W_4 = 12.$$

For either machine in this model it is not possible to analyze redundancy because the PRAM does not model any overhead. It only models load imbalance.