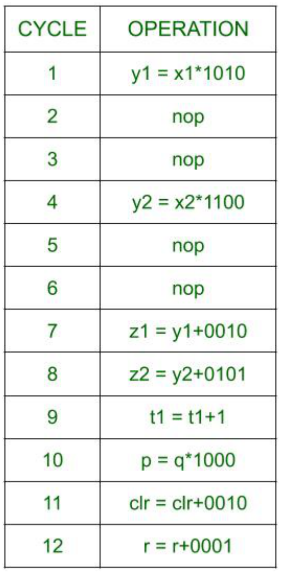
Instruction Level Parallelism (ILP) is used to refer to the architecture in which multiple operations can be performed parallelly in a particular process, with its own set of resources – address space, registers, identifiers, state, program counters. It refers to the compiler design techniques and processors designed to execute operations, like memory load and store, integer addition, float multiplication, in parallel to improve the performance of the processors. Examples of architectures that exploit ILP are VLIWs, Superscalar Architecture.
ILP processors have the same execution hardware as RISC processors. The machines without ILP have complex hardware which is hard to implement. A typical ILP allows multiple-cycle operations to be pipelined.

Example :

Suppose, 4 operations can be carried out in single clock cycle. So there will be 4 functional units, each attached to one of the operations, branch unit, and common register file in the ILP execution hardware. The sub-operations that can be performed by the functional units are Integer ALU, Integer Multiplication, Floating Point Operations, Load, Store. Let the respective latencies be 1, 2, 3, 2, 1.
Let the sequence of instructions be –

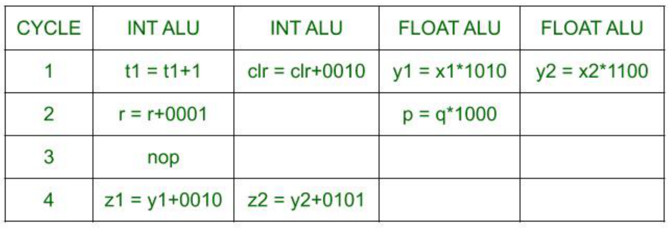
1. y1 = x1*1010
2. y2 = x2*1100
3. z1 = y1+0010
4. z2 = y2+0101
5. t1 = t1+1
6. p = q*1000
7. clr = clr+0010
8. r = r+0001

Sequential record of execution vs. Instruction-level Parallel record of execution –



CYCLE	OPERATION
1	y1 = x1*1010
2	nop
3	nop
4	y2 = x2*1100
5	nop
6	nop
7	z1 = y1+0010
8	z2 = y2+0101
9	t1 = t1+1
10	p = q*1000
11	clr = clr+0010
12	r = r+0001

Fig. a



CYCLE	INT ALU	INT ALU	FLOAT ALU	FLOAT ALU
1	t1 = t1+1	clr = clr+0010	y1 = x1*1010	y2 = x2*1100
2	r = r+0001		p = q*1000	
3	nop			
4	z1 = y1+0010	z2 = y2+0101		

Fig. b

Fig. a shows sequential execution of operations.
Fig. b shows use of ILP in improving performance of the processor.

The ‘nop’s or the ‘no operations’ in the above diagram are used to show idle time of processor. Since latency of floating-point operations is 3, hence multiplications take 3 cycles and processor has to remain idle for that time period. However, in Fig. b processor can utilize those nop’s to execute other operations while previous ones are still being executed.

While in sequential execution, each cycle has only one operation being executed, in processor with ILP, cycle 1 has 4 operations, cycle 2 has 2 operations. In cycle 3 there is ‘nop’ as the next two operations are dependent on first two multiplication operations. The sequential processor takes 12 cycles to execute 8 operations whereas processor with ILP takes only 4 cycles.

Architecture :

Instruction Level Parallelism is achieved when multiple operations are performed in single cycle, that is done by either executing them simultaneously or by utilizing gaps between two successive operations that is created due to the latencies.
Now, the decision of when to execute an operation depends largely on the compiler rather than hardware. However, extent of compiler’s control depends on type of ILP architecture where information regarding parallelism given by compiler to hardware via program varies. The classification of ILP architectures can be done in the following ways –

1. **Sequential Architecture :**
   Here, program is not expected to explicitly convey any information regarding parallelism to hardware, like superscalar architecture.
2. **Dependence Architectures :**
   Here, program explicitly mentions information regarding dependencies between operations like dataflow architecture.
3. **Independence Architecture :**
   Here, program gives information regarding which operations are independent of each other so that they can be executed instead of the ‘nop’s.

In order to apply ILP, compiler and hardware must determine data dependencies, independent operations, and scheduling of these independent operations, assignment of functional unit, and register to store data.