CAO: Lecture 21
Parallelism

Topics Covered

- Goals of Parallelism
- Exploitation of Concurrency
- Types of Parallelism
- Instruction Pipeline
- Four segment CPU Pipeline
- Timing of Instruction Pipeline
- Pipeline Conflicts
- Instruction-level parallelism (ILP)
- Processor Level Parallelism

- Executing two or more operations at the same time is known as Parallelism.

Goals of Parallelism:

- The purpose of parallel processing is to speedup the computer processing capability or in words, it increases the computational speed.
- Increases throughput, i.e. amount of processing that can be accomplished during a given interval of time.
- Improves the performance of the computer for a given clock speed.
- Two or more ALUs in CPU can work concurrently to increase throughput.
- The system may have two or more processors operating concurrently.

Exploitation of Concurrency

Techniques of Concurrency:

- Overlap : execution of multiple operations by heterogenous functional units.
- Parallelism : execution of multiple operations by homogenous functional units.

Throughput Enhancement

A computer's performance is measured by the time taken for executing a program.

The program execution involves performing instruction cycles, which includes two types of operations:

- Internal Micro-operations: performed inside the hardware functional units such as the processor, memory, I/O etc.
- Transfer of information: between different functional hardware units for Instruction fetch, operand fetch, I/O operation etc.

Types of Parallelism:

Instruction Level Parallelism (ILP)

- Pipelining
- Superscalar

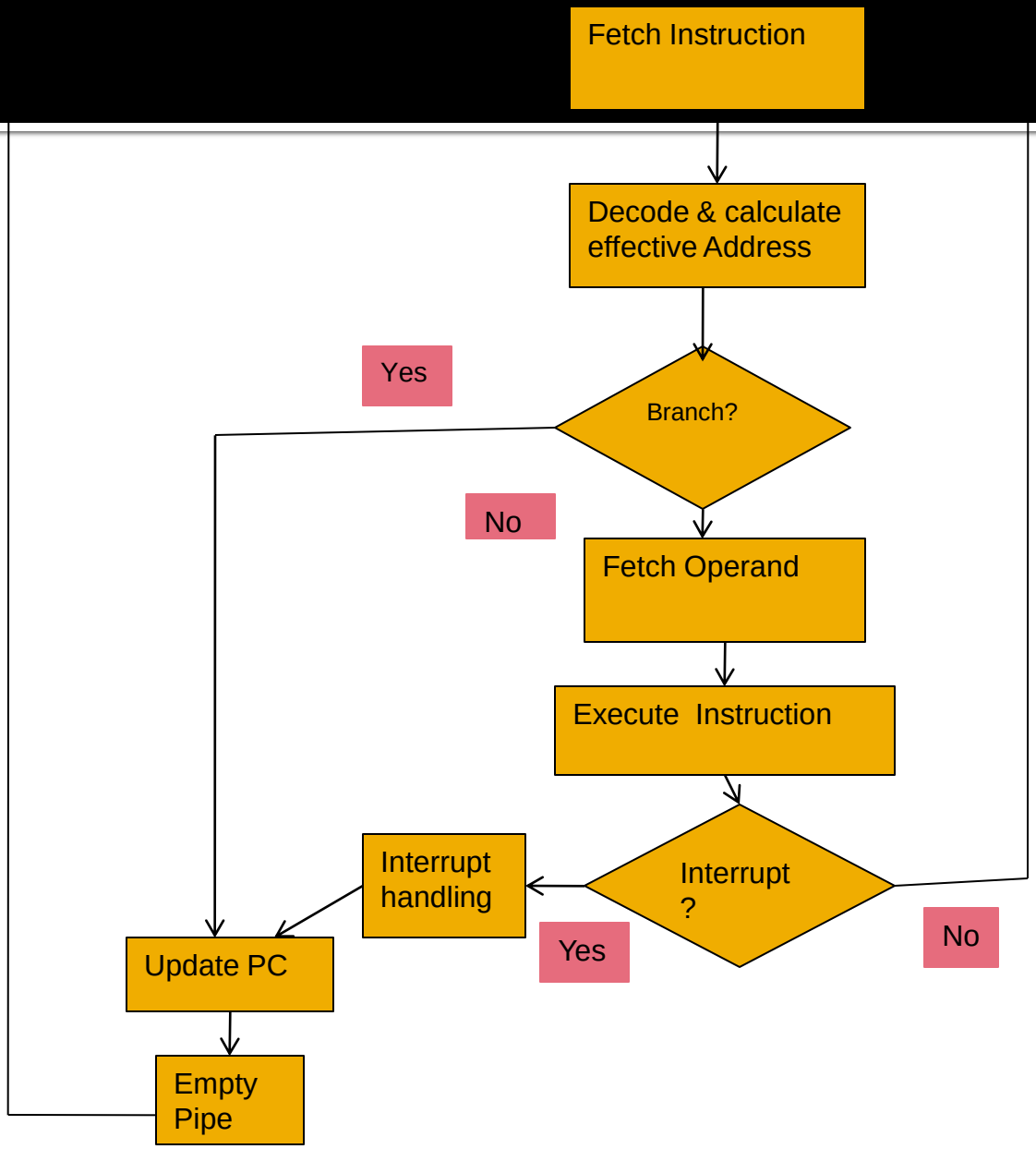
Processor Level Parallelism

- Array Computer
- Multiprocessor

Instruction Pipeline

- An instruction pipeline reads consecutive instructions from memory while previous instructions are being executed in other segments.
- Computer needs to process each instruction with the following sequence of steps.
  - Fetch the instruction from memory
  - Decode the instruction
  - Calculate the effective address
  - Fetch the operands from memory
  - Execute the instruction
  - Store the result in the proper place

Four segment CPU Pipeline



Timing of Instruction Pipeline

Instruction	Step 1	2	3	4	5	6	7	8	9	10	11	12	13
1	FI	DA	FO	EX									
2		FI	DA	FO									
3			FI	DA									
4				FI	-	-	FI	DA	FO	EX			
5					-	-	-	FI	DA	FO	EX		
6									FI	DA	FO	EX	
7										FI	DA	FO	EX

Pipeline Conflicts

- Resource conflicts caused by access to memory by two segments at the same time. These may be resolved by using separate instruction and data memories.
- Data Dependency conflicts arise when an instruction depends on the result of a previous instruction, but this result is not yet available.
- Branch Difficulties arise from branch and other instructions that change the value of PC.

Instruction-level parallelism (ILP)

- **Instruction-level parallelism** (ILP) is a measure of how many of the operations in a computer program can be performed simultaneously.

- Micro-architectural techniques that are used to exploit ILP include:
  - Instruction pipelining where the execution of multiple instructions can be partially overlapped.

  - Superscalar execution in which multiple execution units are used to execute multiple instructions in parallel. In typical superscalar processors, the instructions executing simultaneously are adjacent in the original program order.

- A **superscalar** CPU architecture implements a form of parallelism called instruction-level parallelism within a single processor.
- It therefore allows faster CPU throughput than would otherwise be possible at a given clock rate.
- A superscalar processor executes more than one instruction during a clock cycle by simultaneously dispatching multiple instructions to redundant functional units on the processor.
- Each functional unit is not a separate CPU core but an execution resource within a single CPU such as an arithmetic logic unit, a bit shifter, or a multiplier.
- While a superscalar CPU is typically also pipelined.
- Pipelining and Superscalar architecture are considered different performance enhancement techniques.

The superscalar technique is associated with several identifying characteristics (within a given CPU core):

- Instructions are issued from a sequential instruction stream.
- CPU hardware dynamically checks for data dependencies between instructions at run time (versus software checking at compile time)
- The CPU accepts multiple instructions per clock cycle.

- Available performance improvement from superscalar techniques is limited by two key areas:
- The degree of intrinsic parallelism in the instruction stream, i.e. limited amount of instruction-level parallelism, and
- The complexity and time cost of the dispatcher and associated dependency checking logic.
- The branch instruction processing.

Processor Level Parallelism

- **Multiprocessing** is the use of two or more central processing units (CPUs) within a single computer system.
- The term also refers to the ability of a system to support more than one processor and/or the ability to allocate tasks between them.
- *Multiprocessing* sometimes refers to the execution of multiple concurrent software processes in a system as opposed to a single process at any one instant.
- The terms multitasking or multiprogramming are more appropriate to describe this concept, which is implemented mostly in software, whereas multiprocessing is more appropriate to describe the use of multiple hardware CPUs.
- A system can be both multiprocessing and multiprogramming, only one of the two, or neither of the two.
- In a **multiprocessing** system, all CPUs may be equal, or some may be reserved for special purposes.

- In multiprocessing, the processors can be used to execute a single sequence of instructions in multiple contexts
- In a single instruction stream, single data stream or SISD, one processor sequentially processes instructions, each instruction processes one data item.
- Single-instruction, multiple-data or SIMD, often used in vector processing
- Multiple sequences of instructions in a single context multiple-instruction, single-data or MISD, used to describe pipelined processors.
- Multiple sequences of instructions in multiple contexts (multiple-instruction, multiple-data or MIMD.