Chapter 06: Instruction Pipelining and Parallel Processing

Lesson 09:

Superscalar Processors and Parallel Computer Systems

Objective

- To understand parallel pipelines and multiple execution units
- Instruction level parallelism in superscalar processors

Multiple execution Units in Superscalar Processor

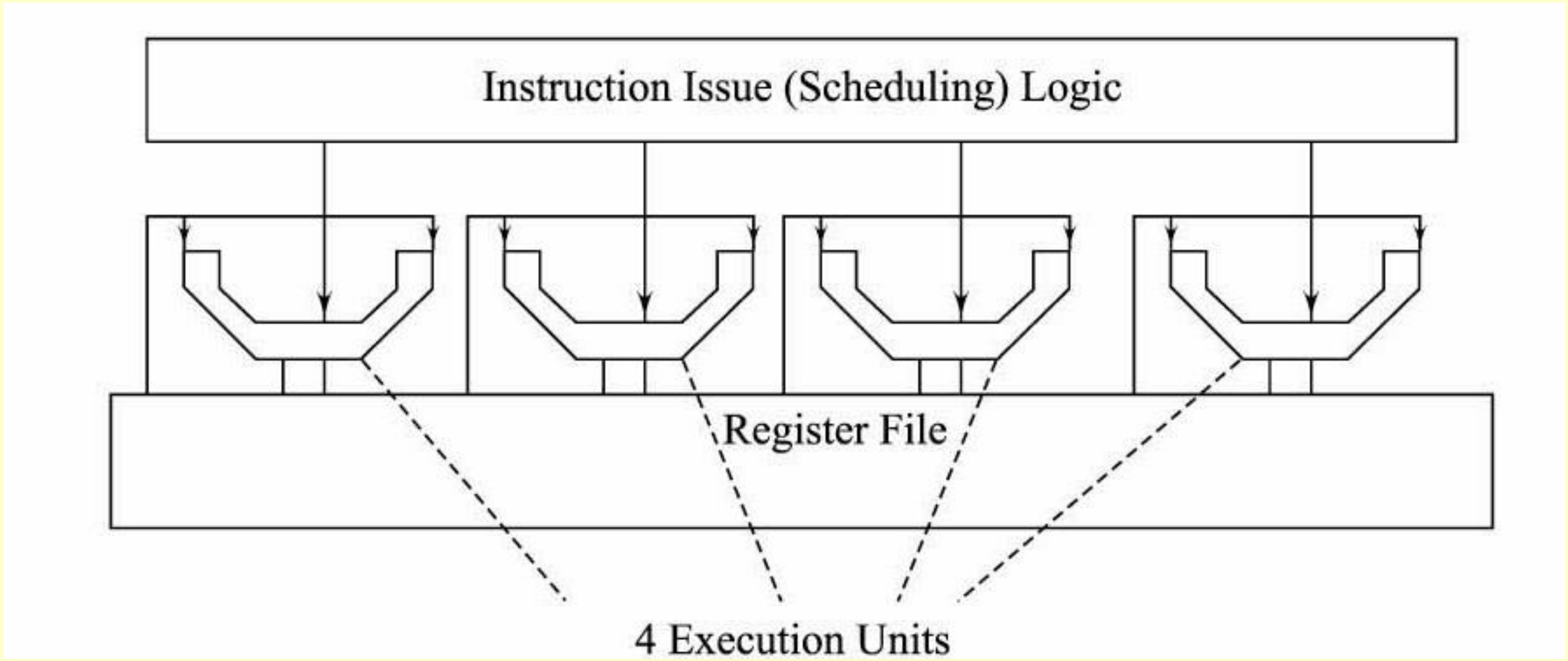
Superscalar Processor

- Multiple execution units to execute instructions
- Each execution unit reads its operands from and writes its results to a single, centralized register file

Multiple execution Units - Superscalar Processor

- When an operation writes its result back to the register file, that result becomes visible to all of the execution units on the next cycle, allowing operations to execute on different units from the operations that generate their inputs

Instruction Issue Logic and Four execution units in a superscalar



Instruction Level Parallelism (ILP) in Superscalar processors

- Have complex bypassing hardware that forwards the results of each instruction to all of the execution units to reduce the delay between dependent instructions
- The instructions that make up a program are handled in superscalar processors by the instruction issue logic, which issues instructions to the units in parallel

Superscalar processors Instruction Issue Logic

- Allows control flow changes, such as branches, to occur simultaneously across all of the units, making it much easier to write and compile programs for instruction-level parallel superscalar processors

A superscalar processor hardware

- Extracts instruction-level parallelism from sequential programs
- During each cycle, the instruction issue logic of a superscalar processor examines the instructions in the sequential program to determine which instructions may be issued on that cycle

Instruction level parallelism

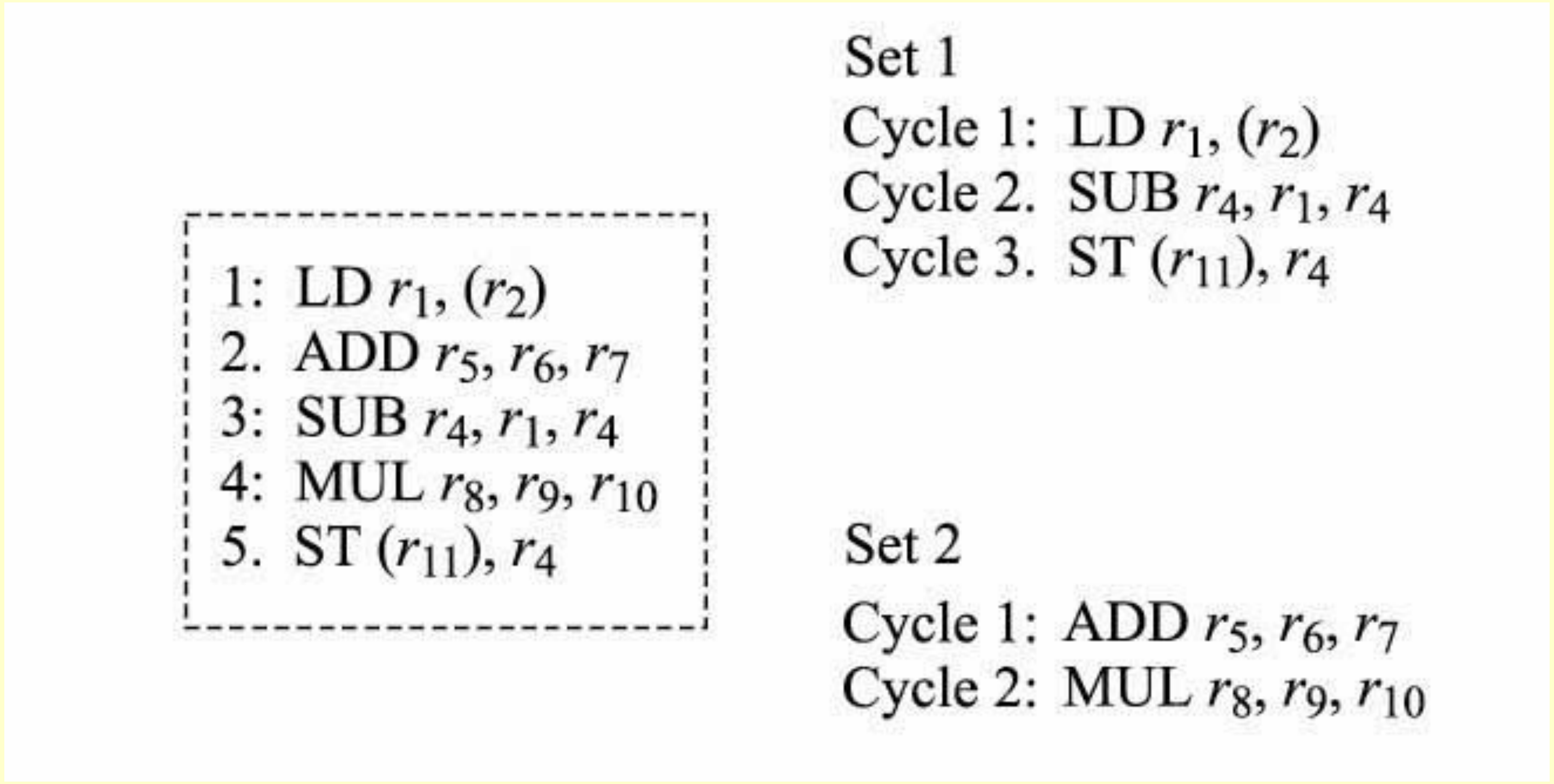
Instruction-level parallelism

- If enough parallelism exists within a program, a superscalar processor can execute one instruction per execution unit per cycle
- Even if the program was originally compiled for execution on a processor that could only execute one instruction per cycle

Instruction-level parallelism

- Instruction-level parallel processors exploit the fact that many of the instructions in a sequential program do not depend on the instructions that immediately precede them in the program

Example of ILP



1: LD r_1, (r_2)
2. ADD r_5, r_6, r_7
3: SUB r_4, r_1, r_4
4: MUL r_8, r_9, r_{10}
5. ST (r_{11}), r_4

Set 1

Cycle 1: LD r_1, (r_2)
Cycle 2. SUB r_4, r_1, r_4
Cycle 3. ST (r_{11}), r_4

Set 2

Cycle 1: ADD r_5, r_6, r_7
Cycle 2: MUL r_8, r_9, r_{10}

Example

- Instructions I1, I3, and I5 are dependent on each other because I1 generates a data value that is used by I3, which generates a result that is used by I5

Example

- I2 and I4 do not use the results of any other instructions in the fragment and do not generate any results that are used by instructions in the fragment

Data Dependencies

- Require that I1, I3, and I5 be executed in order to generate the correct result
- I2 and I4 can be executed before, after, or in parallel with any of the other instructions without changing the results of the program fragment

Latency

- Program fragment in three cycles if each instruction had a latency of one cycle
- Because I1, I3, and I5 are dependent, it is not possible to reduce the execution time of the fragment any further by increasing the number of instructions that the processor can execute simultaneously

Techniques to Overcome ILP processor Performance—Limitations from Data Dependency related Stalls, Branch Penalties, and other factors

Limitations

- RAW hazard (dependencies) limit performance by requiring that instructions be executed in sequence to generate the correct results, and they represent a fundamental limitation on the amount of performance
- Possess hardware to efficiently take care of the collisions in execution unit and of data and control hazards

Limitations

- Instructions with WAW hazard (dependencies) and WAR dependencies
- Branch Penalty Limitations

Limitations

- Processor does not know which instructions will be executed after a branch until the branch has completed
- Branches limitation on instruction-level parallelism
- Requirement of Branch prediction hardware

Hardware Characteristics of superscalar

- Reduces data dependencies (data hazards) and check the stalls efficiently
- Reduces control dependencies (hazards) and branch delays efficiently

Hardware Characteristics

- Select the program, order and process in order but results evaluated later out of order

Hardware reservation station

- Virtual execution unit and register renaming
- Enabling dynamic scheduling (hardware out-of-order reordering) of the instruction to increase the performance by taking care of above limiting factor

Summary

We Learnt

- Example of Parallel Pipelines
- Improvement in Performance
- Limitations and methods to overcome these
- Scheduling the instructions dynamically and by predicting branches by hardware

End of Lesson 09 on

Superscalar Processors and Parallel Computer Systems