Lecture 7

Dr: Alshaimaa Mostafa Mohammed

Chapter 3

Instruction Level Parallelism

Instruction Level Parallelism

- Almost all processors since 1985 use pipelining to overlap the execution of instructions and improve performance.
- This potential overlap among instructions is called **instruction level parallelism**
- We look at a wide range of techniques for extending the basic pipelining concepts by increasing the amount of parallelism exploited among instructions.

Instruction Level Parallelism

- Almost all processors since 1985 use pipelining to overlap the execution of instructions and improve performance.
- There are two largely separable approaches to exploiting ILP:

(1) an approach that relies on hardware to help discover and exploit the parallelism dynamically,

(2) an approach that relies on software technology to find parallelism statically at compile time.

Instruction Level Parallelism

- Processors using the dynamic, hardware-based approach, including the Intel Core series, dominate in the desktop and server markets.
- In the personal mobile device market, where energy efficiency is often the key objective, designers exploit lower levels of instruction-level parallelism.
- First introduced in the IBM Stretch (Model 7030) in about 1959
- Later the CDC 6600 incorporated pipelining and the use of multiple functional units
- The Intel i486 was the first pipelined implementation of the IA32 architecture

Instruction Level Parallelism

- Instruction level parallel processing is the concurrent processing of multiple instructions
- basic block—a straight-line code sequence with no branches in except to the entry and no branches out except at the exit
- The amount of parallelism available within a basic block is quite small.
- Typical MIPS programs have a dynamic branch frequency of between 15% and 25%
  - That is, between three and six instructions execute between a pair of branches, and data hazards usually exist within these instructions as they are likely to be dependent
- Given basic code block size in number of instructions, ILP must be exploited across multiple blocks

Instruction Level Parallelism

- The simplest and most common way to increase the ILP is to exploit parallelism among iterations of a loop. This type of parallelism is often called ***loop-level parallelism***.

Loop Level Parallelism
Exploitation among Iterations of a Loop

- Loop adding two 1000 element arrays
  - Code

```
for (i=1; i<= 1000; i=i+1)
        x[i] = x[i] + y[i];
```

- If we look at the generated code, within a loop there may be little opportunity for overlap of instructions, but each iteration of the loop can overlap with any other iteration

Concepts and Challenges Approaches to Exploiting ILP

- Two major approaches
  - Dynamic – these approaches depend upon the hardware to locate the parallelism
  - Static – fixed solutions generated by the compiler, and thus bound at compile time
- These approaches are not totally disjoint, some requiring both
- Limitations are imposed by data and control hazards

- An important alternative method for exploiting loop-level parallelism is the use of SIMD in both vector processors and Graphics Processing Units (GPUs)
- A SIMD instruction exploits data-level parallelism by operating on a small to moderate number of data items in parallel (typically two to eight).
- A vector instruction exploits data-level parallelism by operating on many data items in parallel using both parallel execution units and a deep pipeline.