Seminar 2

1. Instruction Pipelining

2. Superscalars

3. Parallel Architectures
Problem 1

A nonpipelined processor has a clock rate of 2.5 GHz and an average CPI (cycles per instruction) of 4. An upgrade to the processor introduces a five-stage pipeline. However, due to internal pipeline delays, the clock rate of the new processor has to be reduced to 2 GHz.

a. What is the speedup achieved for a sequence of 100 instructions?
b. What is the MIPS (millions of instructions per second) rate for each processor?

Remember from Lecture 3:

\[ S_{k,n} = \frac{n \times k}{k + (n - 1)} \]

- \( n \): number of instructions to execute
- \( k \): number of pipeline stages
- \( S_{k,n} \): (theoretical) speedup produced by a pipeline with \( k \) stages when executing \( n \) instructions
Solution

a.

\[ S_{5,100} = \frac{100 \times 5}{5 + (100 - 1)} = 4.8 \]

We would have a speedup of 4.8 if the pipelined processor would work at the same clock rate as the initial one!

But the clock rate of the pipelined processor is reduced by a factor of \( \frac{2}{2.5} = 0.8 \)

\[ S = 4.8 \times 0.8 = 3.8 \]

b.

- For the nonpipelined processor:
  
  clock rate = 2.5GHz; 4 clock cycles/instruction \( \Rightarrow \)
  
  \[ 2.5 \times 10^3 / 4 \text{ Million instr/sec} = 625 \text{ MIPS}. \]

- For the pipelined processor:
  
  clock rate = 2 GHz; one instruction completed/clock cycle \( \Rightarrow \)
  
  \[ 2 \times 10^3 \text{ Million instr/sec} = 2000 \text{ MIPS}. \]
Problem 2

Consider the following assembly language program:

I1: Move R3, R7  \hspace{1cm} R3 ← R7
I2: Load R8, (R3)  \hspace{1cm} R8 ← (R3)
I3: Add R3, R3, 4  \hspace{1cm} R3 ← R3 + 4
I4: Load R9, (R3)  \hspace{1cm} R9 ← (R3)
I5: BLE R8, R9, L3  Branch if R9 > R8

Show the dependencies.

Solution

True data dependency (RAW): I1 - I2, I1 - I3; I3 - I4, I2 - I5, I4 - I5

Output dependency (WAW): I1 - I3

Antidependency (WAR): I2 - I3
**Problem 3**

a. Identify the dependencies in the following code:

   I1: R1 ← 100
   I2: R5 ← R1 + R2
   I3: R7 ← R5 + 1
   I4: R1 ← R2 + R4
   I5: R5 ← 0
   I6: R2 ← R4 − 25
   I7: R3 ← R7 − 2
   I8: R4 ← R1 + R3
   I9: R10 ← 0
   I10: R1 ← R1 + 30

b. Rename the registers in the above sequence to prevent, where possible, dependency problems.

c. Consider a superscalar computer on which the execution of each instruction takes one cycle; the computer has two arithmetic units. Show how instructions are executed in consecutive cycles with:
   c1) in order execution;
   c2) out of order execution before renaming;
   c3) out of order execution after renaming.
Solution

a. 
True data dependency (RAW): I1 - I2, I2 - I3, I3 - I7, I4 - I8, I4 - I10, I7 - I8.
Output dependency (WAW): I1 - I4, I4 - I10, I1 - I10, I2 - I5.
Antidependency (WAR): I2 - I4, I2 - I10, I2 - I6, I3 - I5, I4 - I6, I6 - I8, I8 - I10

b. 
I1: R1 ← 100  
I2: R5 ← R1 + R2  
I3: R7 ← R5 + 1  
I4: R11 ← R2 + R4  
I5: R51 ← 0  
I6: R21 ← R4 − 25  
I7: R3 ← R7 − 2  
I8: R41 ← R11 + R3  
I9: R10 ← 0  
I10: R12 ← R11 + 30
### c1: In order

<table>
<thead>
<tr>
<th>Cycle</th>
<th>ALU1</th>
<th>ALU2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>i1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>i2</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>i3</td>
<td>i4</td>
</tr>
<tr>
<td>4</td>
<td>i5</td>
<td>i6</td>
</tr>
<tr>
<td>5</td>
<td>i7</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>i8</td>
<td>i9</td>
</tr>
<tr>
<td>7</td>
<td>i10</td>
<td></td>
</tr>
</tbody>
</table>

### c2: Out of order, no renaming

<table>
<thead>
<tr>
<th>Cycle</th>
<th>ALU1</th>
<th>ALU2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>i1</td>
<td>i9</td>
</tr>
<tr>
<td>2</td>
<td>i2</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>i3</td>
<td>i4</td>
</tr>
<tr>
<td>4</td>
<td>i5</td>
<td>i6</td>
</tr>
<tr>
<td>5</td>
<td>i7</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>i8</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>i10</td>
<td></td>
</tr>
</tbody>
</table>

### c3: Out of order, renaming

<table>
<thead>
<tr>
<th>Cycle</th>
<th>ALU1</th>
<th>ALU2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>i1</td>
<td>i4</td>
</tr>
<tr>
<td>2</td>
<td>i2</td>
<td>i5</td>
</tr>
<tr>
<td>3</td>
<td>i3</td>
<td>i6</td>
</tr>
<tr>
<td>4</td>
<td>i7</td>
<td>i9</td>
</tr>
<tr>
<td>5</td>
<td>i8</td>
<td>i10</td>
</tr>
</tbody>
</table>
Problem 4

An application program is executed on a nine-processor cluster. The program took time $T$ on this cluster. Further, it was found that 25% of $T$ was time in which the application was running simultaneously on all nine processors. The remaining time, the application had to run on a single processor.

a. Calculate the speedup under the aforementioned conditions (relative to execution on a single processor).

b. Suppose that we are able to effectively use 17 processors rather than 9 on the parallelized portion of the code. Calculate the speedup (relative to execution on a single processor) that is achieved.
Solution

a.

\[ S^9 = \frac{T_S}{T_P^9} = \frac{T_S}{T} \]

\[ T_S = 0.25 \times T \times 9 + 0.75 \times T = 3T \quad \implies \quad S^9 = 3 \]

b.

\[ S^{17} = \frac{T_S}{T_P^{17}} = \frac{3T}{T_P^{17}} \]

\[ T_P^{17} = 0.25 \times \frac{9}{17} \times T + 0.75 \times T = 0.88T \]

\[ \downarrow \]

\[ S^{17} = \frac{3T}{0.88T} = 3.4 \]
Problem 5

The following program is to be executed on a processor, and a parallel version is to be executed on a 32-processor cluster:

L1: \textbf{for} (i=0; i<1024; i++){
L2: \quad \text{sum}[i] = 0;
L3: \quad \textbf{for} (j=0; j<=i; j++)
L4 \quad \text{sum}[i] = \text{sum}[i]+i
L5 \}

Suppose lines 2 and 4 each take two machine cycle times, including all processor and memory access activities. Ignore the overhead caused by the software loop control statements (lines 1, 3, 5) and all the system overhead and resource conflicts.

a. What is the total execution time (in machine cycle times) of the program on a single processor?

b. Divide the i-loop iterations among the 32 processors as follows: processor 1 executes the first 32 iterations (i= 0 to 31), processor 2 executes the next 32 iterations, and so on. What are the execution time and speedup factor compared to execution on a single processor.

c. Explain how to modify the parallelizing to facilitate a balanced parallel execution of all the computational workload over 32 processors.

d. What is the minimum execution time resulting from the parallel execution on 32 processors? What is the resulting speedup over a single processor?
Solution

a.

\[ \begin{align*}
  i=0: & \quad L2 + L4 \\
  i=1: & \quad L2 + 2L4 \\
  i=2: & \quad L2 + 3L4 \\
  \vdots
  \end{align*} \]

\[ \begin{align*}
  i=1023: & \quad L2 + 1024L4 \\
  \end{align*} \]

Total: \[ 1024L2 + (1+2+3+ \ldots +1024)L4 = 1024L2 + 524800L4 \]

Duration of L2 and L4: 2 machine cycles

Total duration: \( (1024 + 524800)*2 = 1051648 \) machine cycles
b.

\[ i: 0..31, \text{proc.1: } (L2+L4)+(L2+2L4)+...+(L2+32L4) \]
\[ i: 32..63, \text{proc.2: } (L2+33L4)+(L2+34L4)+...+(L2+64L4) \]

-----------------------------------------

\[ i: 992..1023, \text{pr.32: } (L2+993L4)+(L2+994L4)+...+(L2+1024L4) \]

The overall execution time is determined by processor 32:
\[ 32L2 + (993 + 994 + ... 1024)L4 = 32L2 + 32272L4 \]

\[ \text{Total duration: } (32 + 32272)*2 = 64608 \text{ machine cycles} \]

\[ \text{Speedup} = 1051648/64608 = 16.28 \]
c.

By distributing the i-loops in strict order on each processor, the load is very unbalanced (e.g. processor 1 executes 1120 machine cycles while processor 32 executes 64608 machine cycles).

In order to get a balanced load, the i-loops have to be distributed such that each processor executes: $32L2 + 16400L4$, which results in 32864 machine cycles.

Each processor will execute 32 i-loops as follows:
Processor 1: $i = 0 \ldots 15$ and $i = 1008 \ldots 1023$
Processor 2: $i = 16 \ldots 31$ and $i = 992 \ldots 1007$
Processor 32: $i = 496 \ldots 511$ and $i = 512 \ldots 527$

Total execution time: 32864 cycles.
Speedup = $1051648/32864 = 32$
(the load is perfectly balanced over the 32 processors)