Review of Software Execution (1)

- Recall:
  1. High level code is compiled to ___________ instructions
  2. When executed, CPU uses PC register to know what instruction to ________ next
  3. PC is incremented and used to fetch the next instruction
  4. Registers are used as temp. storage of variables
  5. Variables "live" in ________ & must be updated

#define MAX 1000000;
int data[MAX];

void init()
{
  int i;
  for(i=0; i < MAX; i++){
    data[i] = i;
  }
}

int main() {
  init();
  /* More work */
  return 0;
}

Some details have been left out (how does sf get initialized, incremented, etc.)

Review of Software Execution (2)

- Time 1: Store
- Time 2: Increment i
- Time 3: Loop & repeat
- Time 4

Some details have been left out (how does sf get initialized, incremented, etc.)

Improving Performance

- We want to improve the performance of our computation
- Question: What are we referring to when we say "performance"?
  - ___________
  - ___________
  - ___________
- We will primarily consider __________ in this discussion
How Do We Measure Speed

- **Fundamental Measurement**: _________
  - Absolute time from ______ to _________
  - To compare two alternative systems (HW + SW) and their performance, start a timer when you begin a task and stop it when the task ends
  - Do this for both systems and compare the resulting times
- We call this the _________ of the system and it works great from the perspective of the ___________ task
  - If system A completes the task in 2 seconds and system B requires 3 seconds, then system A is clearly superior
- But when we dig deeper and realize that the single, overall task is likely made of _________ small tasks, we can consider more than just latency

Performance Depends on View Point?!

- What’s faster to get from point A to point B?
  - A 747 Jumbo Airliner
  - An F-22 fighter jet
- If only _____________ to get from point A to point B, then the _________
  - This is known as _____________ [units of seconds]
  - Time from the start of an operation until it completes
- If _____________ to get from point A to point B, the ______ looks much better
  - This is known as _____________ [jobs/second]
- The **overall** execution time (latency) may best be improved by ___________ throughput and not the latency of individual tasks

Hardware Techniques

- We can add hardware or reorganize our hardware to improve throughput and latency of individual tasks in an effort to reduce the total latency (time) to finish the overall task
- We will look at two examples:
  - Caching: Improves _____________
  - Pipelining: Improves _____________

Improving Latency and Throughput

**CACHING AND PIPELINING**
Caching

- **Cache (def.)** — "to store away in hiding or for future use"
- **Primary idea**
  - The ________ you access or use something you expend the ________ amount of time to get it
  - However, store it somewhere (i.e. in a cache) you can get it more ________ the next time you need it
  - The next time you need something check if it is in the cache first
  - If it is in the cache, you can get it quickly; else go get it expending the full amount of time (but then ________ it in the cache)
- **Examples:**
  - __________________
  - __________________
  - __________________

Cache Overview

- Remember what register are used for?
  - Quick access to copies of data
  - Only a ________ (32 or 64) so that we can access really quickly
  - Controlled by the ________
- Cache memory is a small-ish, (____ bytes to a few _____ bytes) "____" memory usually built onto the processor chip
- Will hold ________ of the latest data & instructions accessed by the processor
- Managed by the ________
  - ________ to the software

Cache Operation (1)

- When processor wants data or instructions it always ________ in the cache first
- If it is there, ________ access
- If not, get it from __________
- Memory will also supply ________ data since it is likely to be needed soon
- Why?
  - Things like ________ & ________ (instructions) are commonly accessed sequentially

Cache Operation (2)

- When processor asks for the data again or for the next data value in the array (or instruction of the code) the cache will likely have it
- Questions?

Main point: Caching reduces the latency of memory accesses which improves overall program performance.
### Memory Hierarchy & Caching

- Use several levels of faster and faster memory to hide _______ of larger levels.

![Memory Hierarchy Diagram]

- Registers
- L1 Cache
  - ~ 1 ns
- L2 Cache
  - ~ 10 ns
- Main Memory
  - ~ 100 ns

<table>
<thead>
<tr>
<th>Unit of Transfer</th>
<th>8 - 64 bytes</th>
<th>8 - to 64 bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>Faster</td>
<td>More Expensive</td>
<td>Faster</td>
</tr>
<tr>
<td>Less Expensive</td>
<td>Slower</td>
<td>Slower</td>
</tr>
<tr>
<td>Smaller</td>
<td>More Expensive</td>
<td>Smaller</td>
</tr>
<tr>
<td>Larger</td>
<td>Less Expensive</td>
<td>Larger</td>
</tr>
</tbody>
</table>

### Pipelining

- We'll now look at a hardware technique called **pipelining** to improve ________________.
- The key idea is to __________ the processing of multiple "items" (either data or instructions).

### Example

- Suppose you are asked to build dedicated hardware to perform some operation on all 100 elements of some arrays.
- Suppose the operation \((A[i]+B[i])/4\) takes 10 ns to perform.
- How long would it take to process the entire arrays: _______ ns
  - Can we improve?

```plaintext
for(i=0; i < 100; i++)
C[i] = (A[i] + B[i]) / 4;
```

- Clock Freq. = \(\frac{1}{1250}\) MHz

### Pipelining Example

- Pipelining refers to insertion of registers to split combinational logic into smaller stages that can be overlapped in time (i.e. create an assembly line).

<table>
<thead>
<tr>
<th>Clock Cycle</th>
<th>Stage 1</th>
<th>Stage 2</th>
</tr>
</thead>
</table>

Clock freq: = __________ MHz

- Time for 0th elements to complete: __________
- Time between each of the remaining 99 element completing: __________
- Total: __________
- Define:
  - \(\text{sdegup} = \) __________
  - \(\text{speedup} = \frac{1000 \text{ns}}{\text{speedup}} = __________\)
  - Clock freq: = __________
**Need for Registers**

- Provides separation between combinational functions
  - Without registers, fast signals could “catch-up” to data values in the next operation stage

**Pipelining Example**

- By adding more pipelined stages we can improve throughput
- Have we affected the latency of processing individual elements? __________
- Questions/Issues?
  - __________ stage delays
  - __________ of registers (Not free to split stages)
  - This limits how much we can split our logic

**Non-Pipelined Processors**

- Currently we know our processors execute software 1 instruction at a time
- 3 steps/stages of work for each instruction are:
  - __________
  - __________
  - __________

**Pipelined Processors**

- By breaking our processor hardware for instruction execution into stages we can overlap these stages of work
- Latency for a single instruction is the __________
- Overall throughput, and thus total latency, are greatly improved
More and More Stages

- We can break the basic stages of work into substages to get better performance
- In doing so our clock period goes _____; frequency goes _____
- All kinds of interesting issues come up though when we overlap instructions and our discussed in future CENG courses

Summary

- By investing extra hardware we can improve the overall latency of computation
- Measures of performance:
  - Latency is start to finish time
  - Throughput is tasks completed per unit time (measure of parallelism)
- Caching reduces latency by holding data we will use in the future in quickly accessible memory
- Pipelining improves throughput by overlapping processing of multiple items (i.e. an assembly line)