Review of Computer Architecture

- Credit: Most of the slides are made by Prof. Wayne Wolf who is the author of the textbook.
- I made some modifications to the note for clarity.
- Assume some background information from CSCE 430 or equivalent
von Neumann architecture

- Memory holds data and instructions.
- Central processing unit (CPU) fetches instructions from memory.
  - Separate CPU and memory distinguishes programmable computer.
- CPU registers help out: program counter (PC), instruction register (IR), general-purpose registers, etc.
Recalling Pipelining
Recalling Pipelining

What is a potential Problem with von Neumann Architecture?

Harvard architecture
von Neumann vs. Harvard

- Harvard can’t use self-modifying code.
- Harvard allows two simultaneous memory fetches.
- Most DSPs (e.g., Blackfin from ADI) use Harvard architecture for streaming data:
  - greater memory bandwidth.
  - different memory bit depths between instruction and data.
  - more predictable bandwidth.

Today’s Processors

Harvard or von Neumann?
RISC vs. CISC

- Complex instruction set computer (CISC):
  - many addressing modes;
  - many operations.
- Reduced instruction set computer (RISC):
  - load/store;
  - pipelinable instructions.

Instruction set characteristics

- Fixed vs. variable length.
- Addressing modes.
- Number of operands.
- Types of operands.
Tensilica Xtensa

- RISC based
- variable length
- But not CISC

Programming model

- Programming model: registers visible to the programmer.
- Some registers are not visible (IR).
Multiple implementations

- Successful architectures have several implementations:
  - varying clock speeds;
  - different bus widths;
  - different cache sizes, associativities, configurations;
  - local memory, etc.

Assembly language

- One-to-one with instructions (more or less).
- Basic features:
  - One instruction per line.
  - Labels provide names for addresses (usually in first column).
  - Instructions often start in later columns.
  - Columns run to end of line.
ARM assembly language example

```
label1   ADR r4,c
        LDR r0,[r4] ; a comment
        ADR r4,d
        LDR r1,[r4]
        SUB r0,r0,r1 ; comment

  destination
```

Pseudo-ops

- Some assembler directives don’t correspond directly to instructions:
  - Define current address.
  - Reserve storage.
  - Constants.
Pipelining

- Execute several instructions simultaneously but at different stages.
- Simple three-stage pipe:

```
memory -> fetch -> decode -> execute
```

Pipeline complications

- May not always be able to predict the next instruction:
  - Conditional branch.
- Causes bubble in the pipeline:

```
fetch  decode  Execute
     JNZ
```

```
fetch  decode  execute
fetch  decode  execute
fetch  decode  execute
```
Superscalar

- RISC pipeline executes one instruction per clock cycle (usually).
- Superscalar machines execute multiple instructions per clock cycle.
  - Faster execution.
  - More variability in execution times.
  - More expensive CPU.

Simple superscalar

- Execute floating point and integer instruction at the same time.
  - Use different registers.
  - Floating point operations use their own hardware unit.
- Must wait for completion when floating point, integer units communicate.
Costs

- Good news---can find parallelism at run time.
  - Bad news---causes variations in execution time.
- Requires a lot of hardware.
  - \( n^2 \) instruction unit hardware for \( n \)-instruction parallelism.

Finding parallelism

- Independent operations can be performed in parallel:
  ADD r0, r0, r1
  ADD r3, r2, r3
  ADD r6, r4, r0
Pipeline hazards

- Two operations that have data dependency cannot be executed in parallel:
  \[ x = a + b; \]
  \[ a = d + e; \]
  \[ y = a - f; \]

Order of execution

- In-order:
  - Machine stops issuing instructions when the next instruction can’t be dispatched.

- Out-of-order:
  - Machine will change order of instructions to keep dispatching.
  - Substantially faster but also more complex.
VLIW architectures

- Very long instruction word (VLIW) processing provides significant parallelism.
- Rely on compilers to identify parallelism.

What is VLIW?

- Parallel function units with shared register file:

  ![Diagram of VLIW architecture]

  - register file
  - function unit
  - function unit
  - function unit
  - ... function unit
  - instruction decode and memory
VLIW cluster

- Organized into clusters to accommodate available register bandwidth:

VLIW and compilers

- VLIW requires considerably more sophisticated compiler technology than traditional architectures---must be able to extract parallelism to keep the instructions full.
- Many VLIWs have good compiler support.
Scheduling

EPIC

- EPIC = Explicitly parallel instruction computing.
- Used in Intel/HP Merced (IA-64) machine.
- Incorporates several features to allow machine to find, exploit increased parallelism.
IA-64 instruction format

- Instructions are bundled with tag to indicate which instructions can be executed in parallel:

- Memory system

- CPU fetches data, instructions from a memory hierarchy:
Memory hierarchy complications

- Program behavior is much more state-dependent.
  - Depends on how earlier execution left the cache.
- Execution time is less predictable.
  - Memory access times can vary by 100X.

### Memory Hierarchy Complication

<table>
<thead>
<tr>
<th></th>
<th>Pentium 3-M</th>
<th>Pentium 4-M</th>
<th>Pentium M</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Core</strong></td>
<td>P6 (Tualatin 0.13µ)</td>
<td>Netburst (Northwood 0.13µ)</td>
<td>&quot;P6+&quot; (Banias 0.13µ, Dothan 0.09µ)</td>
</tr>
<tr>
<td><strong>L1 Cache</strong></td>
<td>16Kb + 16Kb</td>
<td>8Kb + 12Kµops (TC)</td>
<td>32Kb + 32Kb</td>
</tr>
<tr>
<td><strong>L2 Cache</strong></td>
<td>512Kb</td>
<td>512Kb</td>
<td>1024Kb</td>
</tr>
<tr>
<td><strong>Instructions Sets</strong></td>
<td>MMX, SSE</td>
<td>MMX, SSE, SSE2</td>
<td>MMX, SSE, SSE2</td>
</tr>
<tr>
<td><strong>Max frequencies</strong> (CPU/FSB)</td>
<td>1.2GHz 133MHz</td>
<td>2.4GHz 400MHz (QDR)</td>
<td>2GHz 400MHz (QDR)</td>
</tr>
<tr>
<td><strong>Number of transistors</strong></td>
<td>44M</td>
<td>55M</td>
<td>77M, 140M</td>
</tr>
<tr>
<td><strong>SpeedStep</strong></td>
<td>2nd generation</td>
<td>2nd generation</td>
<td>3rd generation</td>
</tr>
</tbody>
</table>
End of Overview

- Next class: Altera Nios II processors