Lecture Topics

- Today: Overview
  (P&H 1.1-1.8)

- Next: Integer Arithmetic
  (P&H 3.1-3.4)

Handouts

- Syllabus (old)

Announcements

- Course website
  
  http://www.cse.msu.edu/~cse420/

- Consulting hours

- Accounts and remote access

- Introduction to Sim
System Overview

- **Hardware**
  - Processor
  - Primary memory
  - I/O devices
- **Systems software**
  - Operating system
  - Compiler, assembler, linker
- **Applications software**

Hardware Overview

- **Processor**
  - Datapath (ALU, register file) – performs arithmetic calculations and comparisons
  - Control unit – manages fetch/execute cycle, overall operation of the processor
  - Cache memory (SRAM) – high-speed buffer for primary memory
- **Primary storage (DRAM)**
  - volatile – does not retain data when powered off
Hardware Overview

- Input/output devices
  - User interface (display, keyboard, mouse)
  - Secondary storage (disk, CD/DVD, flash)
  - Network devices

Fetch-Execute Cycle

- The control unit manages the overall execution of the processor through the fetch/execute cycle

Fetch Phase:
- use PC to copy next instruction from primary memory to IR

Execute Phase:
- decode IR, take appropriate action
- update PC
Example: ARM microprocessor

- Assume each instruction is 4 bytes long
- Assume PC: 00010700
- Fetch phase:
  - access primary memory at address 00010700
  - copy 4 bytes into IR
- IR now contains: E0861007

Example (continued)

- IR now contains: E0861007
- Execute phase:
  - decode IR
    - ADD instruction on ARM
  - take appropriate action
  - update PC
    - PC + 4 ==> PC
Example (continued)

<table>
<thead>
<tr>
<th>Address</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>00000000</td>
<td></td>
</tr>
<tr>
<td>00000001</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>00010700</td>
<td>E0</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>FFFFFFFD</td>
<td></td>
</tr>
<tr>
<td>FFFFFFFE</td>
<td></td>
</tr>
<tr>
<td>FFFFFFFF</td>
<td></td>
</tr>
</tbody>
</table>

CPU
PC: 00010700
IR: E0861007

Example (continued)

CPU
PC: 00010700
IR: E0861007

ADD
R[7] →
Example: C/C++ Program

From www.cse.msu.edu/~cse420/Examples/example01

```c
int sum = 0;

int main()
{
    for( int i = 1; i <= 6; i++)
    {
        sum = sum + i;
    }
}
```

ARM Assembly Language

```assembly
.global main
.text
main:    push {lr}
         mov    r0, #1
loop:    cmp     r0, #6
         bgt     end
         ldr     r2, =sum
         ldr     r1, [r2]
         add     r1, r1, r0
         str     r1, [r2]
         add     r0, r0, #1
         b       loop
end:     pop {lr}
         mov     pc, lr
```

ARM Machine Language

.global main
.text
0000 E52DE004      main:   push    {lr}
0004 E3A00001              mov     r0, #1
0008 E3500006      loop:   cmp     r0, #6
000c CA000005              bgt     end
0010 E59F2018              ldr     r2, =sum
0014 E5921000              ldr     r1, [r2]
0018 E0811000              add     r1, r1, r0
001c E5821000              str     r1, [r2]
0020 E2800001      add     r0, r0, #1
0024 EAFFFF77                    b       loop
0028 E49DE004      end:    pop     {lr}
002c E1A0F00E              mov     pc, lr

Execution Trace

<table>
<thead>
<tr>
<th>Time</th>
<th>PC</th>
<th>IR</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>00010800</td>
<td>E52DE004 * main: push {lr}</td>
</tr>
<tr>
<td>1</td>
<td>00010804</td>
<td>E3A00001 mov r0, #1</td>
</tr>
<tr>
<td>2</td>
<td>00010808</td>
<td>E3500006 loop: cmp r0, #6</td>
</tr>
<tr>
<td>3</td>
<td>0001080c</td>
<td>CA000005 bgt end</td>
</tr>
<tr>
<td>4</td>
<td>00010810</td>
<td>E59F2018 * ldr r2, =sum</td>
</tr>
<tr>
<td>5</td>
<td>00010814</td>
<td>E5921000 * ldr r1, [r2]</td>
</tr>
<tr>
<td>6</td>
<td>00010818</td>
<td>E0811000 add r1, r1, r0</td>
</tr>
<tr>
<td>7</td>
<td>0001081c</td>
<td>E5821000 * str r1, [r2]</td>
</tr>
<tr>
<td>8</td>
<td>00010820</td>
<td>E2800001 add r0, r0, #1</td>
</tr>
<tr>
<td>9</td>
<td>00010824</td>
<td>EAFFFF77 b loop</td>
</tr>
<tr>
<td>10</td>
<td>00010808</td>
<td>E3500006 loop: cmp r0, #6</td>
</tr>
<tr>
<td>11</td>
<td>0001080c</td>
<td>CA000005 bgt end</td>
</tr>
</tbody>
</table>
Translation Stages

- **High-level language**
  - Representation closer to problem domain
  - Helps with productivity and portability
- **Assembly language**
  - Textual representation of machine instructions
- **Machine language**
  - Bit patterns

Assessing Performance

- **Goal:** determine which computer (out of several alternatives) is the best to use in a particular situation
- **Must define what is meant by** *performance*
- **Must have basis for comparisons**
Example: Passenger Planes

- Commercial passenger planes

<table>
<thead>
<tr>
<th>Airplane</th>
<th>Passenger capacity</th>
<th>Cruising range (miles)</th>
<th>Cruising speed (m.p.h.)</th>
<th>Passenger throughput (passengers x m.p.h.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boeing 777</td>
<td>375</td>
<td>4630</td>
<td>610</td>
<td>228,750</td>
</tr>
<tr>
<td>Boeing 747</td>
<td>470</td>
<td>4150</td>
<td>610</td>
<td>286,700</td>
</tr>
<tr>
<td>BMC/Sud Concorde</td>
<td>132</td>
<td>4000</td>
<td>1350</td>
<td>178,200</td>
</tr>
<tr>
<td>Douglas DC-8-50</td>
<td>146</td>
<td>8720</td>
<td>644</td>
<td>79,424</td>
</tr>
</tbody>
</table>

- Which definition of performance?
  - capacity?
  - range?
  - speed?

Example (continued)
Assessing Performance

- Response time (execution time): total time required to complete a particular task
  - CPU execution time
  - Memory accesses
  - Disk accesses
  - I/O activities
  - OS overhead

- Throughput (bandwidth): total amount of work done in given amount of time

Assessing Performance

- Users primarily concerned with response time: how long it takes to complete a task
- Managers primarily concerned with throughput: total work done per time unit
- Note that response time and throughput are related: if response time decreases, throughput increases
- Primary focus for next few weeks: response time
Terminology

- Performance: reciprocal of execution time
  \[ \text{performance} = \frac{1}{\text{execution time}} \]
- To increase performance, you must decrease execution time
- P&H: "improve performance" or "improve execution time" means "increase performance" (and "decrease execution time")

Terminology

- Relative performance:
  \[ \frac{\text{Performance}_x}{\text{Performance}_y} = \frac{\text{Execution time}_y}{\text{Execution time}_x} = n \]
- P&H:
  "X is n times faster than Y"
  "X is n times as fast as Y"
Example

- Assume:
  - Computer A runs program in 10 secs
  - Computer B runs program in 15 secs

- What is the relative performance?
  \[
  \frac{15}{10} = 1.5
  \]

- A is 1.5 times faster than B

Measuring Time

- Response time (elapsed time, wall clock time): total time to complete a task, including CPU execution time, time to access memory and disk, time to complete I/O activities, system overhead

- Modern systems use time sharing, so it is often useful to focus only on CPU execution time

- System performance: response time on unloaded system

- CPU performance: CPU execution time
CPU Clocking

- Operations sequenced by constant-rate clock

![Diagram showing CPU clocking with clock period, data transfer, and state update]

Example:

- Clock period = 250ps = 0.25ns = 250 × 10^{-12}s
- Clock frequency = 4.0GHz = 4000MHz = 4.0 × 10^9 Hz

CPU Execution Time

\[
\text{CPU Time} = \frac{\text{CPU Clock Cycles} \times \text{Clock Cycle Time}}{\text{Clock Rate}} = \frac{\text{CPU Clock Cycles}}{\text{Clock Rate}}
\]

- Performance can be improved by
  - Reducing number of clock cycles
  - Increasing clock rate

- Hardware designer must often trade off clock rate against cycle count
Example

- Computer A: 2 GHz clock, 10 sec CPU time
- Designing Computer B: goal is 6 sec CPU time
  - Can use faster clock, but causes $1.2 \times$ clock cycles
- How fast must Computer B clock be?

\[
\text{Clock Rate}_B = \frac{\text{Clock Cycles}_B}{\text{CPU Time}_B} = \frac{1.2 \times \text{Clock Cycles}_A}{6 \text{ sec}}
\]
\[
\text{Clock Cycles}_A = \text{CPU Time}_A \times \text{Clock Rate}_A
\]
\[
= 10 \text{ sec} \times 2 \text{ GHz} = 20 \times 10^9
\]
\[
\text{Clock Rate}_B = \frac{1.2 \times 20 \times 10^9}{6 \text{ sec}} = \frac{24 \times 10^9}{6 \text{ sec}} = 4 \text{ GHz}
\]

Instruction Count and CPI

\[
\text{Clock Cycles} = \text{Instruction Count} \times \text{Cycles per Instruction}
\]
\[
\text{CPU Time} = \text{Instruction Count} \times \text{CPI} \times \text{Clock Cycle Time}
\]
\[
= \frac{\text{Instruction Count} \times \text{CPI}}{\text{Clock Rate}}
\]

- Instruction Count for a program is determined by program, ISA and compiler
- Average clock cycles per instruction is determined by CPU hardware
CPI Example

- Computer A: Cycle Time = 250ps, CPI = 2.0
- Computer B: Cycle Time = 500ps, CPI = 1.2
- Same ISA
- Which is faster, and by how much?

\[
\begin{align*}
\text{CPU Time}_A &= \text{Instruction Count} \times CPI_A \times \text{Cycle Time}_A \\
&= I \times 2.0 \times 250 \text{ ps} = I \times 500 \text{ ps} \quad \text{A is faster…} \\
\text{CPU Time}_B &= \text{Instruction Count} \times CPI_B \times \text{Cycle Time}_B \\
&= I \times 1.2 \times 500 \text{ ps} = I \times 600 \text{ ps} \\
\text{CPU Time}_B &= I \times 600 \text{ ps} \\
\text{CPU Time}_A &= I \times 500 \text{ ps} = 1.2 \\
\end{align*}
\]

…by this much

CPI in More Detail

- If different instruction classes take different numbers of cycles

\[
\text{Clock Cycles} = \sum_{i=1}^{n} (\text{CPI}_i \times \text{Instruction Count}_i)
\]

- Weighted average CPI

\[
\text{CPI} = \frac{\text{Clock Cycles}}{\text{Instruction Count}} = \sum_{i=1}^{n} \left( \frac{\text{CPI}_i \times \text{Instruction Count}_i}{\text{Instruction Count}_i} \right)
\]

Relative frequency
CPI Example

- Alternative code sequences using instructions in classes A, B, C

<table>
<thead>
<tr>
<th>Class</th>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPI for class</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>IC in sequence 1</td>
<td>2</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>IC in sequence 2</td>
<td>4</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Sequence 1: IC = 5
- Clock Cycles
  - \(= 2 \times 1 + 1 \times 2 + 2 \times 3\)
  - \(= 10\)
- Avg. CPI = \(10/5 = 2.0\)

Sequence 2: IC = 6
- Clock Cycles
  - \(= 4 \times 1 + 1 \times 2 + 1 \times 3\)
  - \(= 9\)
- Avg. CPI = \(9/6 = 1.5\)

Performance Summary

\[
\text{CPU Time} = \frac{\text{Instructions}}{\text{Program}} \times \frac{\text{Clock cycles}}{\text{Instruction}} \times \frac{\text{Seconds}}{\text{Clock cycle}}
\]

- Performance depends on
  - Algorithm: affects IC, possibly CPI
  - Programming language: affects IC, CPI
  - Compiler: affects IC, CPI
  - Instruction set architecture: affects IC, CPI, \(T_c\)