CS-1000 An Introduction to Computer Architecture

Dr. Soner Onder
Michigan Tech
October 13, 2015
About Me

- BSc degree in Chemical Engineering from METU, Ankara, Turkey.
- MSc in Computer Engineering, METU, Ankara, Turkey.
- PhD in Computer Science, University of Pittsburgh, PA.
- Worked in industry both as a systems programmer, as well as a field engineer (8+ Years) before starting my PhD.
- Developed thousands of lines of code, most of which were utilized heavily.
About Me

- Married to one of your professors.
- Have two kids, one is majoring in Chemical Engineering, the other is a sophomore in Calumet High School.
- Have a furry orange tabby cat
  - He probably is wearing a costume (not sure).
  - Acts like a black cat in Halloween.

That is not him!
My view of Computer Science

Without Algorithms and Theory there is NO Computer Science.

Without SYSTEMS, there is NO MACHINE (i.e., Computer).

Without COMPUTER, there is no Smart Phone!

What is **SYSTEMS**?

- The core is Computer Architecture.
- Programming Languages and Compilers.
- Operating Systems.
- Computer Networks.
Intel 4004 (1971)

Maximum clock rate was 740 kHz.
Instruction cycle time: 10.8 µs.
(8 clock cycles / instruction cycle)
46300 to 92600 instructions per second.

Adding two 8-digit numbers (32 bits each, assuming 4-bit BCD digits) was stated as taking 850 µs - i.e. 79 instruction cycles, about 10 instruction cycles per decimal digit.

Instruction set contained 46 instructions (of which 41 were 8 bits wide and 5 were 16 bits wide)

Register set contained 16 registers of 4 bits each
Intel Core Architecture (2006)

Clock rate 3GHz.
6 - 9 Billion Instructions per second.

L1 cache 64 kB per core
L2 cache 1 MB to 8 MB unified
L3 cache 8 MB to 16 MB shared (Xeon)
Transistors 105M 65 nm
Intel Core i7 (2008)

Clock rate 3-3.5 GHZ.
6 -9 Billion Instructions per second/CPU.

Transistors
- 730M 45 nm
- 1.8 B (6 core – 2013)
- 5.560 B (18-core Xeon Haswell- 2014)

Outlook:
- 100 B transistors in 2020 !
Is Computer Architecture Circuits (Hardware)?

• No. But you need to understand how the hardware works.
• It is how we put together circuits (at a higher level of abstraction, and algorithmically):
  – Intel Core i7 (single core) / Intel 4004 =
  – 3,000,000,000 (Hz) / 740,000 (Hz) = 4054 times faster.
  – 6,000,000,000 (instructions/sec) / 92600 (instructions/sec) = 64,795
  – A factor of roughly 16 in performance!
• That is the power of computer architecture:
  – Modern processors process multiple instructions per cycle
  – They act speculatively to mitigate delays
  – They use sophisticated algorithms to efficiently execute programs.
Computer Architecture

Computer Architecture is a core field of computer science which sits at the cross-roads of abstractions.

- Very vibrant field – needs always changing together with opportunities.
  - New circuit techniques enable new architectures.
  - New architectures may facilitate new techniques.
- Optimize for power, performance (or both).

- Computer Architecture can potentially impact everything (yes, you can also save the world by being an architect!)
- Very high paying (and satisfying) good jobs too ..
  - Processors are everywhere from simple machines to war planes, from factories to kitchen appliances.
Revisiting Computer Science

- It is the science of creating and utilizing abstractions to achieve computation.
- Using abstractions is the only way we know to create complex systems.

- **Computer Architecture is a core field of computer science which sits at the cross-roads of abstractions.**
  
  - *Only if you learn and understand all the layers we use in computation you can become a good architect.*
Layers of Abstractions

1. Problem
2. Algorithm
3. Language (Program)
4. Instructions (ISA)
5. Micro-Architecture
6. Circuit
7. Electrons

Software ?
Compiler
Hardware ?

SYSTEMS !

Computer Architecture
My Research

• Primarily concentrated on three fronts:
  1. Seeking alternative forms of execution models so that sequential programs can be executed efficiently by highly parallel architectures.
  2. Dealing with latency/delays: Seeking ways to execute dependent instructions together.
  3. Applying AI techniques on Computer Architecture, primarily on simulators to verify their correctness and further understand behavior of complex architectures.
This is a joint four year project between MTU and FSU

(Co-PIs : Soner Onder and David Whalley)

Funded by NSF ( $745,000, MTU Share $560,000, MTU is the lead institution).

Project Goals:

- Exploit both regular and irregular parallelism.
- Massive ILP through LaZy execution.
- Imperative programming languages by translating to FGSA.
- Single-assignment form for both the compiler and the architecture.
- Multi-core uniprocessor!
An FGSA Example

Algorithms for converting programs into FGSA:

Dr. Shuhan Ding
PhD in 2012
Michigan Tech

CC = <<{B1.x,B2.x}, {B3.x,B4.x}, {p, ¬ p}, ψ>
Supporting Execution Models - the old shoe

do i=0 step 1 until n
sum = sum + a[i];
print sum;

ρ₁ = true
ρ₂ = true
x₀ = 0
y₀ = a
k₀ = n << 2
m₀ = y₀ + k₀

x₁ = ψ₁ ρ₁(x₀, x₂)
y₁ = ψ₂ ρ₂(y₀, y₂)
z₀ = M[y₁]
x₂ = x₁ + z₀
y₂ = y₁ + 4
p = y₂ <= m₀
if (p)

x₃ = η ¬p(x₂)
print x₃

<table>
<thead>
<tr>
<th>n</th>
<th>a[0]</th>
<th>a[1]</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10</td>
<td>20</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ρ₁</th>
<th>ρ₂</th>
<th>x₀</th>
<th>y₀</th>
<th>k₀</th>
<th>m₀</th>
</tr>
</thead>
<tbody>
<tr>
<td>false</td>
<td>false</td>
<td>0</td>
<td>a</td>
<td>4</td>
<td>a+4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>x₁</th>
<th>y₁</th>
<th>z₀</th>
<th>x₂</th>
<th>y₂</th>
<th>p</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>a+4</td>
<td>20</td>
<td>30</td>
<td>a+8</td>
<td>false</td>
</tr>
</tbody>
</table>

x₃
30
Supporting Execution Models – demand driven execution

do i=0 step 1 until n
sum = sum + a[i];
print sum;

\[ \begin{align*}
\rho_1 &= \text{true} \\
\rho_2 &= \text{true} \\
x_0 &= 0 \\
y_0 &= a \\
k_0 &= n \ll 2 \\
m_0 &= y_0 + k_0 \\
\end{align*} \]

\[ \begin{align*}
x_1 &= \psi_{\rho_1}(x_0, x_2) \\
y_1 &= \psi_{\rho_2}(y_0, y_2) \\
z_0 &= M[y_1] \\
x_2 &= x_1 + z_0 \\
y_2 &= y_1 + 4 \\
p &= y_2 \leq m_0 \\
\end{align*} \]

\[ \begin{align*}
x_3 &= \eta_{\sim p}(x_2) \\
\text{print } x_3
\end{align*} \]

\begin{array}{cccccc}
\text{n} & \text{a[0]} & \text{a[1]} \\
\hline
1 & 10 & 20 \\
\end{array}

\[ \begin{align*}
\rho_1 & \quad \rho_2 & \quad x_0 & \quad y_0 & \quad k_0 & \quad m_0 \\
\text{false} & \text{false} & 0 & a & 4 & a+4 \\
\end{align*} \]

\[ \begin{align*}
x_1 & \quad y_1 & \quad z_0 & \quad x_2 & \quad y_2 & \quad p & \quad x_3 \\
0 & a & 10 & 10 & a+4 & \text{true} \\
\end{align*} \]

Demand

\[ \begin{align*}
x_3 \\
p, x_2 \\
y_2, m_0, x_1, z_0 \\
y_1, y_0, k_0, \rho_1 \\
\rho_2 \\
\rho_1, k_0, y_0 \\
x_0 \\
\rho_2 \\
x_0, m_0, y_1 \\
z_0, y_2, x_1 \\
p, x_2 \\
\end{align*} \]

End of first iteration. \( \eta \) sees that \( \sim p \) is false and demands both \( p \) and \( x_2 \) again.