ECE 411
Spring 2016

Lecture 1

Course Introduction
What you can expect to get out of this class:

- to understand fundamental concepts in computer architecture and how they impact application performance and energy efficiency.
- to become confident in programming for performance, scalability, and efficiency.
- to be able to understand and evaluate architectural descriptions of even today’s most complex processors.
- to gain experience designing a working CPU completely from scratch.
- to learn experimental techniques used to evaluate advanced architectural ideas.
Which is faster?

for (i=0; i<N; i=i+1)
    for (j=0; j<N; j=j+1) {
        r = 0;
        for (k=0; k<N; k=k+1)
            r = r + y[i][k] * z[k][j];
        x[i][j] = r;
    }

for (jj=0; jj<N; jj=jj+B)
    for (kk=0; kk<N; kk=kk+B)
        for (i=0; i<N; i=i+1) {
            for (j=jj; j<min(jj+B-1,N); j=j+1)
                r = 0;
            for (k=kk; k<min(kk+B-1,N); k=k+1)
                r = r + y[i][k] * z[k][j];
            x[i][j] = x[i][j] + r;
        }

Significantly Faster!
Which is faster?

load R1, addr1
store R1, addr2
add R0, R2 -> R3
subtract R4, R3 -> R5
add R0, R6 -> R7
store R7, addr3

load R1, addr1
add R0, R2 -> R3
add R0, R6 -> R7
store R1, addr2
subtract R4, R3 -> R5
store R7, addr3

Twice as fast on some machines and same on others
Which is faster?

loop1: add ...
    load ...
    add ...
    bne R1, loop1

loop2: add ...
    load ...
    bne R2, loop2

Identical performance on several machines
Processor Design is Hard/Interesting!

Modern processors have more than a billion transistors!
Processor Design is Hard/Interesting!

FDIV Pentium bug cost 500 million dollars.

Very recently, power density of a processor higher than a nuclear reactor!
Power Efficiency Challenges

- power/energy-constrained computing devices
  - mobile: 1~5W power budget w/ 5Whr battery capacity

1997 (850KW) **Teraflops**  201x (5W)
Classic Dennard’s Scaling

- 0.7× transistor feature size scaling per generation
  - 2.8× more chip capability at the same power
  - 1.4× faster trs
  - 2× more trs
  - 0.7× capacitance
  - 0.7× voltage
End of Dennard’s Scaling

- 0.7× transistor feature size scaling per generation
  - 1.4× more chip capability at the same power

Diagram:
- 2× more trs
- 0.7× capacitance
Computing Landscape & Challenges

- diminishing performance of **general-purpose CPU-based** system
  - power constraint
    - acceleration using CPU+**GPU** (GPGPU computing)
- memory capacity and bandwidth constraints
  - capacity using NVM-base DDRx-T or **LR-DIMM**
Emerging Apps & Characteristics

**traditional apps**

- complex/irregular ops
- large # of ops per value
- high temporal locality

**emerging apps**

- simple repetitive ops
- small # of ops per value
- low-temporal locality
Today

- Course information & structure
- What is this course about?
- What you can expect to learn?

- Data Path Design for an LC3b Processor
Contact Information

- Instructor: Nam Sung Kim
  217 CSL
  Email: nskim@illinois.edu (start subject line with [ECE411])
  Office Hours: 5:00PM-6:00PM, Wed

- TA’s:
  Jones, Russell  rjones27@illinois.edu
  Lu, Dao        daolu1@illinois.edu
  Yan, Yan       yanyan3@illinois.edu
  Zhuge, Chuanhao zhuge2@illinois.edu

Office Hours: posted on course web site:
https://courses.engr.illinois.edu/ece411/index.php
Course Materials

- Primary Textbook: *Computer Organization and Design: The Hardware/Software Interface*
  - Lectures will not always follow the book directly

- Supplementary Materials:
  - Selected notes from Prof. Kim
  - Computer Architecture books on reserve at Grainger
Contact

- Web site (https://courses.engr.illinois.edu/ece411/index.php)
  - Lecture notes, handouts, labs etc
  - Announcements
- Piazza (piazza.com/illinois/spring2016/ece411: also, link from the website)
  - Posting clarification questions
  - General forum to communicate with students, TAs, instructor about aspects of the course.
  - Must join by the end of the day
Workload and Assignments

- 4 Machine Problems
  - MP0, MP1, and MP2 done individually
  - MP3 done in teams of 3 (design project – 3 checkpoints)
  - 45% of final grade (5%, 5%, 10%, 25%)

- 2 Tests + Final
  - 55% of final grade (14%, 14%, 25%)

- Subjective: 2% (class participation, etc.)

- Study Problems
  - Ungraded, learning tool for you
“Curved” but

- In principle, everybody can make an “A+”
- Low grades are evaluated on an individual basis, but be vigilant about where you stand early on

- There will be some opportunities for extra credit primarily through MP3 and design competition
- Design Competition
Academic Integrity

- You are encouraged to discuss assignments with anyone you feel may be able to help you (except during exams)
- Everything you turn in must be your work or that of your team
- Encouraged collaboration:
  - Verbal discussion of problems/solutions
  - Diagrams, notes, other communication aids
- Unacceptable
  - Copying/exchanging of program/SystemVerilog code or text by any means
  - Giving/receiving help on exams, or using any notes/aid beyond those explicitly permitted
Course overview & Syllabus

- Online...
The Von-Neumann Model
Computer Organization

Processor

Memory

I/O

address

data

instructions
Computer Organization

- Executes instructions, which are represented as bit patterns with specific fields interpreted differently
- Contains a register file, which holds data that will be referenced by instructions
- Has a program counter that holds the address of the instruction being executed
- Can access memory
  - Use address to select the location we want to access
  - Random-access: time to access a piece of data is independent of which address the data is stored in
- IO
  - keyboard, mouse, display, network
## Levels of Transformation

<table>
<thead>
<tr>
<th>Task or Application</th>
<th>Algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>High-level Language Program</td>
<td>Machine Language (ISA)</td>
</tr>
<tr>
<td>Microarchitecture</td>
<td>Logic</td>
</tr>
<tr>
<td>Circuits</td>
<td>Devices</td>
</tr>
</tbody>
</table>
Levels of transformation: from problem spec to results

- **High Level Language Program**
- **Compiler**
- **Assembly Language Program**
- **Assembler**
- **Machine Language Program**
- **Machine Interpretation**

```
temp = v[k];
v[k] = v[k+1];
v[k+1] = temp;
```

```
lw $15, 0($2)
lw $16, 4($2)
sw $16, 0($2)
sw $15, 40($2)
```

```
10001100011000100000000000000000
10001100111100010000000000001000
10101100111100010000000000000000
10101100011000100000000000000000
```

```
```
The Instruction Execution Cycle

**Instruction Fetch**
- Obtain instruction from program storage

**Instruction Decode**
- Determine required actions and instruction size

**Operand Fetch**
- Locate and obtain operand data

**Execute**
- Compute result value or status

**Result Store**
- Deposit results in storage for later use

**Next Instruction**
- Determine successor instruction
LC-3b Instruction Set Encodings

- Instructions represented as 16-bit words
- Opcode always high four bits of the word
  - This is one case where the LC-3b is much more simplistic than commercial ISAs, which generally have several different opcode formats for different types of instructions
- Small number of instruction formats
  - Format = mapping of bits in the instruction to operands/outputs/etc.
  - Commercial ISAs often have many more formats
LC-3b Instruction Encoding Examples

- **ADD, AND (without Immediate)**

- **ADD, AND (with Immediate), NOT**
Basic Computer Organization – more details

**Diagram:**
- **I/O**
- **Processor**
  - General-Purpose Registers
  - Control Logic
  - ALUs
  - MAR
  - MDR
  - PC
  - CC
- **Memory**
  - Data
  - Program
Datapath: ALUs Plus Registers

- MAR
- PC
- IR
- Register File
- ALU
- MDR
Register Transfer Level Notation for Fetch

- One line per clock cycle
- Tells you how data moves between registers
- Can have multiple operations in parallel

RTL for the instruction fetch:

```
MAR <- PC; PC <- PC + 2;
MDR <- MEM[MAR];
IR <- MDR;
```
Fetching Instructions

MAR <- PC; PC <- PC + 2;
Fetching Instructions

MDR <- MEM[MAR];
Fetching Instructions

IR <- MDR;

Memory

LoadMAR

MAR

LoadPC

PC

+2

LoadIR

IR

Register File

ALU

LoadMDR

Memory

IR <- MDR;
LC-3b Instruction Encoding Examples

- **ADD, AND (without Immediate)**

  ![Diagram of ADD, AND (without Immediate)]

- **ADD, AND (with Immediate), NOT**

  ![Diagram of ADD, AND (with Immediate), NOT]
RTL For ADD/AND Instructions without Immediate

\[
\text{MAR} \leftarrow \text{PC}; \quad \text{PC} \leftarrow \text{PC} + 2; \\
\text{MDR} \leftarrow \text{MEM}[\text{MAR}]; \\
\text{IR} \leftarrow \text{MDR}; \\
\text{R[DR]} \leftarrow \text{R[SR1]} \text{ OP R[SR2]}; \quad \text{genCC};
\]
Arithmetic Instructions: AND, ADD, NOT

R[DR] <- R[SR1] OP R[SR2]; genCC

Diagram showing the flow of operations from memory to ALU output, including MAR, PC, IR, Register File, ALU, and CC units.
Example 2

- **LD, LDI, LDB**

  - LDB R4, R2, #-5; R4 <- mem[R2-5]

- **BR**

  - LDB R4, R2, #-5; R4 <- mem[R2-5]
RTL For LD, ST

**LD**

MAR <- PC; PC <- PC + 2;
MDR <- MEM[MAR];
IR <- MDR;
MAR <- R[IR[8:6]] + ADJ6(IR[5:0]);
MDR <- MEM[MAR];
R[IR[11:9]] <- MDR; genCC;

**ST**

MAR <- PC; PC <- PC + 2;
MDR <- MEM[MAR];
IR <- MDR;
MAR <- R[IR[8:6]] + ADJ6(IR[5:0]);
MDR <- R[IR[11:9]];
MEM[MAR] <- MDR;
Refined Data Path for Memory: LD

\[ \text{MAR} \leftarrow R[\text{IR}[8:6]] + \text{ADJ6}(\text{IR}[5:0]); \]

\[ \text{MDR} \leftarrow \text{MEM}[\text{MAR}]; \]

\[ R[11:9] \leftarrow \text{MDR}; \text{genCC}; \]
RTL For BR

MAR <- PC; PC <- PC + 2;
MDR <- MEM[MAR];
IR <- MDR;
If ((n AND N) OR (z AND Z) OR (p AND P))
   PC <- PC + ADJ9(IR[8:0]);
Refined Data Path for Control: BR
MP0

- A tutorial
- Assignment available at ECE411 website
- Need to use EW Lab, 57 Grainger and 2022 ECEB
- Start working on MP0 today!