Microprocessors

- Microprocessors are at core of any computing system
Computers

- Where is the microprocessor?

Operation

- What does a computer/microprocessor do?
Operation

- Microprocessor performs computations
  - Input data is processed
  - Processing instructions (program) determines operations
  - Output is result of computation

Microprocessor Components

- Main components of a microprocessor
  - Computation
    » Arithmetic operations
    » Logic operations
    » Moving data
    » Conditional execution
  - Storage
    » Registers
    » Memory
    » Disk
  - I/O
    » File input/output
    » Keyboard/mouse input
    » Network input/output
    » Video output
    » Audio input/output
Example Microprocessors

- **Microcontroller**
  - Digital thermostat, key fob, remote control, digital clock, battery charger, etc.

- **Embedded microprocessor**
  - Cell phone, PDA, wireless router

- **Microprocessor**
  - Laptop, desktop, game console

- **Multiple microprocessors**
  - High-end laptop, desktop, game console, graphics card, supercomputer

- **Many systems have multiple different processors**
  - Microprocessor, hard disk controller, network interface processor, graphics processor, etc.

Atmel ATmega 128

- **Microcontroller**
  - Single processor
  - Instruction set
    - 8-bit RISC
  - Speed
    - Clock: 16MHz
  - Memory
    - Data: 4kB
    - Instruction: 128kB
  - Size
    - A few mm²
  - Power consumption
    - Around 125mW
Intel Core i7

- Microprocessor
  - 6-core processor
  - Instruction set
    » 64-bit CISC
  - Speed
    » Clock: 3.46GHz
  - Memory
    » On-chip cache: 12MB
  - Size
    » 239mm²
    (32nm process)
  - Power consumption:
    » Max 130W

NVIDIA GeForce GTX Titan Black

- Graphics processing unit
  - 2880-core processor
  - Instruction set
    » 64-bit CUDA
  - Speed
    » Clock: 889MHz
  - Size of chip
    » 561mm²
    (28nm process)
    » 7 billion transistors
  - Power consumption:
    » Max 250W
Microcontroller vs. laptop microprocessor

- Same program: count number of prime numbers in interval

```java
public class CountPrimes {
    public static void main(String[] args) {
        final int STEP=100;
        int c,i,n,p;
p=0;
c=100;
for (n=2; n<=2000000; n++) {
    for (i=2; i<=n/2; i++) {
        if (n%i==0)
            break;
    }
    if (i>n/2) {
p++;
    }
    if (n==c) {
        System.out.println("number of primes in [0.."+n+" ] is "+p);
        c+=STEP;
    }
}
}
```

### Arduino Uno

- Code:

```java
int c,i,n,p;
in0:
c+=STEP;
p=0;
fork=1; c=10000000; n++; {
    for (i=2; i<=n/2; i++) {
        if (n%i==0)
            break;
    }
    if (i>n/2) {
p++;
    }
    if (n==c) {
        Serial.print("Number of primes in [0.. ");
        Serial.print(" is ");
        Serial.print(" ");
        Serial.print(" ");
c=10000000;
    }
}
```

- Output:

- number of primes in [0..9200] is 1140
- number of primes in [0..9300] is 1151
- number of primes in [0..9400] is 1162
- number of primes in [0..9500] is 1177
- number of primes in [0..9600] is 1184
- number of primes in [0..9700] is 1197
- number of primes in [0..9800] is 1208
- number of primes in [0..9900] is 1220
- number of primes in [0..10000] is 1229

### Laptop

- Code:

```java
int c,i,n,p;
otin0:
c+=STEP;
p=0;
fork=1; n=00000000; n++; {
    for (i=2; i<=n/2; i++) {
        if (n%i==0)
            break;
    }
    if (i>n/2) {
p++;
    }
    if (n==c) {
        System.out.println("Number of primes in [0.."+n+" ] is "+p);
        c+=STEP;
    }
}
```

- Output:

- number of primes in [0..9200] is 1140
- number of primes in [0..9300] is 1151
- number of primes in [0..9400] is 1162
- number of primes in [0..9500] is 1177
- number of primes in [0..9600] is 1184
- number of primes in [0..9700] is 1197
- number of primes in [0..9800] is 1208
- number of primes in [0..9900] is 1220
- number of primes in [0..10000] is 1229
They are All the Same

- **Same functionality**
  - All processors can compute the same
  - “Universal computer” (Turing-complete instruction set)

- **Difference in performance**
  - Faster execution time of single program
  - Higher processing throughput on multiple programs

- **Software can be compiled for different processors**
  - Depends on software development environment

Alan Turing

- **Alan Turing (1912–1954)**
  - Mathematician/computer scientist
  - Famous for cryptanalysis
    - Helped in breaking German Enigma code during World War II
  - Famous for work on computability
    - Turing machine as “computer”
    - Showed what class of problems can be solved by a computer
  - “Turing Award” highest recognition in the field of computer science
Increasing Performance

• How to increase processing performance?

  – Faster computation
    » Higher clock rate
  – More computation
    » More instructions per cycle
    » Wider words (e.g., 64-bit instead of 32-bit)
  – Data closer to processor
    » Larger on-chip memories
  – Faster I/O
    » Higher memory bandwidth
  – Better program
    » Optimizing compiler
  – Parallel system
    » Multiple processors in parallel
  – …

• Performance improvements come at cost
  • Larger chip, higher power consumption, …
Performance Considerations

- How much performance can be gained?
- Each improvement targets specific component
  - Performance gain limited to much component is used
- Amdahl’s law
  - Improvement affects fraction $P$ of computation time
  - Improvement provides speed up of $S$ (i.e., $S$ times faster)
  - Overall speedup from improvement:
    $$\frac{1}{(1 - P) + \frac{P}{S}}$$
- Example
  - Speed up 30% of instructions ($P=0.3$) by factor 2 ($S=2$)
  - Overall speedup: $1/(0.7 + 0.3/2) = 1.18$ (=18% faster)
- “Make the common case fast”

Exercise

- Program execution time
  - 50% processing
  - 30% memory access
  - 20% disk access
- Improvement options (not quite realistic)
  - Processor upgrade
    » Clock rate from 2.8 GHz to 3.1 GHz ($S=1.1$, $\$200$)
  - Memory upgrade
    » Bus speed from DDR3-800 to DDR3-1333 ($S=1.67$, $\$500$)
  - Disk upgrade
    » Disk type from hard disk to SSD ($S=1.4$, $\$300$)
- Optimization
  - Which improvement provides most speedup?
  - Which improvement provides most speedup per dollar?
Exercise

- Answers

<table>
<thead>
<tr>
<th>Improvement</th>
<th>P</th>
<th>S</th>
<th>cost</th>
<th>overall speedup</th>
<th>speedup/cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>processor</td>
<td>0.5</td>
<td>1.1</td>
<td>200</td>
<td>1.05</td>
<td>0.0052</td>
</tr>
<tr>
<td>memory</td>
<td>0.3</td>
<td>1.67</td>
<td>500</td>
<td>1.14</td>
<td>0.0023</td>
</tr>
<tr>
<td>disk</td>
<td>0.3</td>
<td>1.4</td>
<td>300</td>
<td>1.09</td>
<td>0.0036</td>
</tr>
</tbody>
</table>

- Which improvement provides most speedup?
  - Memory improvement (overall speedup of 1.14)

- Which improvement provides most speedup per dollar?
  - Processor improvement (overall speedup of 0.0052/$)

- Why do computers keep getting faster?

Moore’s Law

- Milestone paper by Gordon Moore:
  - Maximum number of transistors on an integrated circuit roughly doubles every two years

![Image of Moore's Law paper](image-url)
Moore’s Law

- Moore’s law over 50 years:

![Diagram showing the growth of transistors over 50 years with different product generations such as the Intel 4004, 8086, Pentium, etc.]

Side Note: Geek Jokes in the 60’s

- Illustration in Moore’s paper
  - “Outrageous” prediction on what the future will look like
    - Selling “handy home computers” in a department store...
Parallel Processing

- Most modern computers use multiple parallel cores
  - Divide processing across multiple processors
  - Overall processing can be faster
- Challenges
  - Coordination between processors necessary
  - In many cases, coordination may dominate processing time
- Workloads suitable for parallel processing
  - Scientific computing
  - Graphics processing
  - Etc.
- Most current systems are heading toward multi-core
  - Need ways to parallelizing workload (e.g., compiler)

Speedup through Parallelism

- Speedup is gain in processing time
  - Speedup for \( p \) processor cores is \( S_p = \frac{T_1}{T_p} \)
    » \( T_1 \) is processing time on 1 core
    » \( T_p \) is processing time on \( p \) cores
- What trends do you expect for speedup?
Speedup through Parallelism

- Speedup is gain in processing time.
  - Speedup for \( p \) processor cores is \( S_p = \frac{T_1}{T_p} \)
    - \( T_1 \) is processing time on 1 core
    - \( T_p \) is processing time on \( p \) cores
- What trends do you expect for speedup?

```
<table>
<thead>
<tr>
<th>speedup</th>
<th>0</th>
<th>10</th>
<th>20</th>
<th>30</th>
<th>40</th>
<th>50</th>
<th>60</th>
<th>70</th>
<th>80</th>
<th>90</th>
<th>100</th>
</tr>
</thead>
<tbody>
<tr>
<td>number of processors</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>
```

Ideal parallelism

Limited parallelism (ideal)

Limited parallelism (realistic)

no parallelism

```
<table>
<thead>
<tr>
<th>speedup</th>
<th>0</th>
<th>10</th>
<th>20</th>
<th>30</th>
<th>40</th>
<th>50</th>
<th>60</th>
<th>70</th>
<th>80</th>
<th>90</th>
<th>100</th>
</tr>
</thead>
<tbody>
<tr>
<td>number of processors</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>
```

Speedup through Parallelism

- Speedup of multi-threaded (parallel) prime count
Courses in ECE Curriculum

- ECE 232 – Hardware Organization & Design
- ECE 354 – Computer Systems Lab II
- ECE 415/416 – Senior Design Project
- ECE 568 – Computer Architecture I
- ECE 668 – Computer Architecture II
- ECE 669 – Parallel Computer Architecture

Upcoming...

- Next Wednesday: solar cells
  - Power generation
- Moodle quiz