Chapter 1
Introduction
Architecture & Organization 1

• Architecture is those attributes visible to the programmer
  — Instruction set, number of bits used for data representation, I/O mechanisms, addressing techniques.
  — e.g. Is there a multiply instruction?

• Organization is how features are implemented
  — Control signals, interfaces, memory technology.
  — e.g. Is there a hardware multiply unit or is it done by repeated addition?
Architecture & Organization 2

- All Intel x86 family share the same basic architecture
- The IBM System/370 family share the same basic architecture

- This gives code compatibility ... at least backwards
- Organization differs within members of the same family, e.g. floating point numerical co-processors with names like 8087, 80287 and 80387. With very few exceptions, the 80486 and subsequent x86 processors then integrated this x87 functionality on chip.
Structure & Function

- Structure is the way in which components relate to each other
- Function is the operation of individual components as part of the structure
Function

General computer functions:
- Data processing
- Data storage
- Data movement
- Control
Operations (a) Data movement

I/O (peripherals directly attached)
Communications/Networking
(communication lines)

Example application?
Operations (a) Data movement

I/O (peripherals directly attached) Communications/Networking (communication lines)

Camera attached to a PC, sending the frames to a window on the screen of the same PC.
Operations (b) Storage

Example application?
Playing an mp3 file stored in memory to earphones attached to the same PC.
Operation (c) Processing from/to storage

Example application?
Operation (c) Processing from/to storage

Any number-crunching application that takes data from memory and stores the result back in memory.
Operation (d)
Processing from storage to I/O

Example application?
Operation (d)
Processing from storage to I/O

Receiving packets over a network interface, verifying their CRC, then storing them in memory.
Structure - Top Level

- Computer
  - Main Memory
  - Input Output
  - Systems Interconnection
  - Central Processing Unit

Peripherals

Communication lines
Structure - The CPU

CPU
- Registers
- Arithmetic and Login Unit
- Internal CPU Interconnection
- Control Unit

Computer
- I/O
- System Bus
- Memory

CPU Interconnection

I/O
System Bus
Memory
Structure - The Control Unit

- The Control Unit
- CPU
- ALU
- Internal Bus
- Registers
- Control Unit

Control Unit

- Sequencing Login
- Control Unit Registers and Decoders
- Control Memory
Chapter 2
Computer Evolution and Performance
ENIAC - background

- Electronic Numerical Integrator And Computer
- Eckert and Mauchly
- University of Pennsylvania
- Trajectory tables for weapons
- Started 1943
- Finished 1946
  - Too late for war effort
- Used until 1955
ENIAC - details

- Decimal (not binary)
- 20 accumulators of 10 digits
- Programmed manually by switches
- 18,000 vacuum tubes
- 30 tons
- 15,000 square feet
- 140 kW power consumption
- 5,000 additions per second
von Neumann/Turing

- Stored Program concept
- **Main memory storing programs and data**
- ALU operating on binary data
- Control unit interpreting instructions from memory and executing
- Input and output equipment operated by control unit
- Princeton Institute for Advanced Studies (IAS) computer
- Completed 1952
Structure of von Neumann machine
To do in NOTEBOOK for next time

Section 1.3
- Know the definitions of all Key Terms
- Answer all review questions

Section 2.1
- Be able to draw the IAS/von Neumann diagram – fig. 2.1
IAS - details

1000 x 40 bit words, each word representing
  - One 40-bit binary number
  - Two 20-bit instructions:
    - 8 bits opcode
    - 12 bits address

Set of registers (storage in CPU)
  - Memory Buffer Register
  - Memory Address Register
  - Instruction Register
  - Instruction Buffer Register
  - Program Counter
  - Accumulator
  - Multiplier Quotient
John von Neumann and the IAS machine, 1952
IAS organization

Memory Buffer Register either sends data to or receives data from Mem. or I/O

Memory Address Register specifies which Mem. location will be read or written next

Control signals are set by the opcode part of the instruction bits. Examples:
- Bring a new instruction from Mem. (fetch)
- Perform an addition (execute)
Why do we need both a register and a buffer register to hold instructions?

Why does the arrow between PC and MAR point both ways?
IAS organization

Hint: 2 instructions are stored in each memory word

Hint: the next instruction can be found either sequentially, or through a branch (jump)
IAS – the FETCH-EXECUTE cycle

Each instruction is executed in the same two-step manner:

• **FETCH** → load the binary code of the instr. from Memory (or IBR)
  — Opcode goes into IR
  — Address goes into MAR

• **EXECUTE** → send appropriate control signals to do what the instr. needs to do
IAS FETCH-EXECUTE cycle
# IAS – instruction set (architecture!)

<table>
<thead>
<tr>
<th>Instruction Type</th>
<th>Opcode</th>
<th>Symbolic Representation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data transfer</td>
<td>00001010</td>
<td>LOAD MQ</td>
<td>Transfer contents of register MQ to the accumulator AC</td>
</tr>
<tr>
<td>Data transfer</td>
<td>00001001</td>
<td>LOAD MQ, M(X)</td>
<td>Transfer contents of memory location X to MQ</td>
</tr>
<tr>
<td>Data transfer</td>
<td>00100001</td>
<td>STOR M(X)</td>
<td>Transfer contents of accumulator to memory location X</td>
</tr>
<tr>
<td>Data transfer</td>
<td>00000001</td>
<td>LOAD M(X)</td>
<td>Transfer M(X) to the accumulator</td>
</tr>
<tr>
<td>Data transfer</td>
<td>00000010</td>
<td>LOAD –M(X)</td>
<td>Transfer –M(X) to the accumulator</td>
</tr>
<tr>
<td>Data transfer</td>
<td>00000011</td>
<td>LOAD</td>
<td>M(X)</td>
</tr>
<tr>
<td>Data transfer</td>
<td>00000100</td>
<td>LOAD –</td>
<td>M(X)</td>
</tr>
<tr>
<td>Unconditional branch</td>
<td>00001101</td>
<td>JUMP M(X,0:19)</td>
<td>Take next instruction from left half of M(X)</td>
</tr>
<tr>
<td>Unconditional branch</td>
<td>00001110</td>
<td>JUMP M(X,20:39)</td>
<td>Take next instruction from right half of M(X)</td>
</tr>
<tr>
<td>Conditional branch</td>
<td>00001111</td>
<td>JUMP+ M(X,0:19)</td>
<td>If number in the accumulator is nonnegative, take next instruction from left half of M(X)</td>
</tr>
<tr>
<td>Conditional branch</td>
<td>00100000</td>
<td>JUMP+ M(X,20:39)</td>
<td>If number in the accumulator is nonnegative, take next instruction from right half of M(X)</td>
</tr>
</tbody>
</table>

Specifies one of 21 instructions

There was no assembly language back then!
### IAS – instruction set (continued)

<table>
<thead>
<tr>
<th>Code</th>
<th>Instruction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>00000101</td>
<td>ADD M(X)</td>
<td>Add M(X) to AC; put the result in AC</td>
</tr>
<tr>
<td>00000111</td>
<td>ADD</td>
<td>M(X)</td>
</tr>
<tr>
<td>00000110</td>
<td>SUB M(X)</td>
<td>Subtract M(X) from AC; put the result in AC</td>
</tr>
<tr>
<td>00001000</td>
<td>SUB</td>
<td>M(X)</td>
</tr>
<tr>
<td>00001011</td>
<td>MUL M(X)</td>
<td>Multiply M(X) by MQ; put most significant bits of result in AC, put least significant bits in MQ</td>
</tr>
<tr>
<td>00001100</td>
<td>DIV M(X)</td>
<td>Divide AC by M(X); put the quotient in MQ and the remainder in AC</td>
</tr>
<tr>
<td>00010100</td>
<td>LSH</td>
<td>Multiply accumulator by 2; i.e., shift left one bit position</td>
</tr>
<tr>
<td>00010101</td>
<td>RSH</td>
<td>Divide accumulator by 2; i.e., shift right one bit position</td>
</tr>
<tr>
<td>00010010</td>
<td>STOR M(X,8:19)</td>
<td>Replace left address field at M(X) by 12 rightmost bits of AC</td>
</tr>
<tr>
<td>00010011</td>
<td>STOR M(X,28:39)</td>
<td>Replace right address field at M(X) by 12 rightmost bits of AC</td>
</tr>
</tbody>
</table>
Programming the IAS

• Write a program to add two numbers stored in memory at addresses 0 and 1 and store the result in memory at address 2

• To do in notebook for next time:
  — Write a program that compares the numbers stored in memory at addresses 0 and 1 and places the larger one at address 2
Why are there 12 bits for the memory address?
Commercial Computers

- 1947 - Eckert-Mauchly Computer Corporation
- **UNIVAC I** (Universal Automatic Computer)
- US Bureau of Census 1950 calculations
- Became part of Sperry-Rand Corporation
- Late 1950s - UNIVAC II
  - Faster
  - More memory
IBM

- Punched-card processing equipment
- 1953 - the 701
  - IBM’s first stored program computer
  - Scientific calculations
- 1955 - the 702
  - Business applications
- Lead to 700/7000 series
Second generation of computers
Transistors

- Replaced vacuum tubes
- Smaller
- Cheaper
- Less heat dissipation
- Solid State device
- Made from Silicon (Sand)
- Invented 1947 at Bell Labs by William Shockley et al.
Transistor Based Computers

- NCR & RCA produced small transistor machines
- IBM 7000
- DEC (Digital Equipment Corporation) was founded in 1957
  - Produced PDP-1 in the same year
Third generation of computers: Integrated Circuits

- A computer is made up of gates, memory cells and interconnections
- All these can be manufactured either separately (discrete components) or
- on the same piece of semiconductor (a.k.a. silicon wafer)
Generations of Computers

• Vacuum tube - 1946-1957
• Transistor - 1958-1964
• Small scale integration - 1965 on
  — Up to 100 devices on a chip
• Medium scale integration - to 1971
  — 100-3,000 devices on a chip
• Large scale integration - 1971-1977
  — 3,000 - 100,000 devices on a chip
• Very large scale integration - 1978 -1991
  — 100,000 - 100,000,000 devices on a chip
• Ultra large scale integration – 1991 -
  — Over 100,000,000 devices on a chip
Moore’s Law

- Increased density of components on chip
- Gordon Moore – co-founder of Intel
- Number of transistors on a chip will double every year
- Since 1970’s development has slowed a little
  - Number of transistors doubles every 18 months
- Cost of a chip has remained almost unchanged
- Higher packing density means shorter electrical paths, giving higher performance
- Smaller size gives increased flexibility
- Reduced power and cooling requirements
- Fewer interconnections increases reliability
Growth in CPU Transistor Count
IBM 360 series

- 1964
- Replaced (& not compatible with) 7000 series
- First planned “family” of computers
  - Similar or identical instruction sets
  - Similar or identical O/S
  - Increasing speed
  - Increasing number of I/O ports (i.e. more terminals)
  - Increased memory size
  - Increased cost
- Multiplexed switch structure
DEC PDP-8

- 1964
- First minicomputer (after miniskirt!)
- Did not need air conditioned room
- Small enough to sit on a lab bench
- $16,000
  - $100k+ for IBM 360
- Embedded applications & OEM
- BUS STRUCTURE
DEC - PDP-8 Bus Structure
Semiconductor Memory

- 1970
- Fairchild
- Size of a single core
  - i.e. 1 bit of magnetic core storage
- Holds 256 bits
- Non-destructive read
- Much faster than core
- Capacity approximately doubles each year
To do for next time

• Read from text for next time: Ch. 2 until Microprocessors (p.36)
• Take notes in notebook!
1-minute Quiz

• What is a word in a computer? (p.20)

• Do you think the word size pertains to computer architecture or to computer organization? Explain!
1-minute Quiz

• The computer industry’s first planned family of computers was:
  — The Sperry-Rand UNIVAC
  — The IBM System/360
  — The DEC PDP
  — The IBM System/7000
1-minute Quiz

What was each of these computer’s claim to fame?

- The Sperry-Rand UNIVAC I
- The IBM System/360
- The DEC PDP-8
- The IBM 701
1-minute Quiz

- The text states that the latest semiconductor memories store 16 Giga-bits on a single chip (it’s the binary Giga, $2^{30}$)
- Assume that the word size is one Byte
- How many address lines are needed?
Write a program that compares the numbers stored in memory at addresses 0 and 1 and places the larger one at address 2.
Programming the IAS

Write a program that compares the numbers stored in memory at addresses 0 and 1 and places the larger one at address 2

- Remember: In a von Neumann architecture, both data and program are in the same memory!
- Let’s start storing the program at address 42 (decimal)
- Data segment and code segment – draw the memory map!
Microprocessors - Intel

- **1971 - 4004**
  - First microprocessor
  - All CPU components on a single chip
  - 4 bit
  - Multiplication by repeated addition, no hardware multiplier!

- **Followed in 1972 by 8008**
  - 8 bit
  - Both designed for specific applications

- **1974 - 8080**
  - Intel’s first general purpose microprocessor
# 1970s Processors

<table>
<thead>
<tr>
<th></th>
<th>4004</th>
<th>8008</th>
<th>8080</th>
<th>8086</th>
<th>8088</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clock speeds</td>
<td>108 kHz</td>
<td>108 kHz</td>
<td>2 MHz</td>
<td>5 MHz, 8 MHz, 10 MHz</td>
<td>5 MHz, 8 MHz</td>
</tr>
<tr>
<td>Bus width</td>
<td>4 bits</td>
<td>8 bits</td>
<td>8 bits</td>
<td>16 bits</td>
<td>8 bits</td>
</tr>
<tr>
<td>Number of transistors</td>
<td>2,300</td>
<td>3,500</td>
<td>6,000</td>
<td>29,000</td>
<td>29,000</td>
</tr>
<tr>
<td>Feature size (µm)</td>
<td>10</td>
<td>6</td>
<td>3</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>Addressable memory</td>
<td>640 Bytes</td>
<td>16 KB</td>
<td>64 KB</td>
<td>1 MB</td>
<td>1 MB</td>
</tr>
</tbody>
</table>
1980s Processors

<table>
<thead>
<tr>
<th></th>
<th>80286</th>
<th>386TM DX</th>
<th>386TM SX</th>
<th>486TM DX CPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clock speeds</td>
<td>6 MHz - 12.5 MHz</td>
<td>16 MHz - 33 MHz</td>
<td>16 MHz - 33 MHz</td>
<td>25 MHz - 50 MHz</td>
</tr>
<tr>
<td>Bus width</td>
<td>16 bits</td>
<td>32 bits</td>
<td>16 bits</td>
<td>32 bits</td>
</tr>
<tr>
<td>Number of transistors</td>
<td>134,000</td>
<td>275,000</td>
<td>275,000</td>
<td>1.2 million</td>
</tr>
<tr>
<td>Feature size (µm)</td>
<td>1.5</td>
<td>1</td>
<td>1</td>
<td>0.8 - 1</td>
</tr>
<tr>
<td>Addressable memory</td>
<td>16 MB</td>
<td>4 GB</td>
<td>16 MB</td>
<td>4 GB</td>
</tr>
<tr>
<td>Virtual memory</td>
<td>1 GB</td>
<td>64 TB</td>
<td>64 TB</td>
<td>64 TB</td>
</tr>
<tr>
<td>Cache</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>8 kB</td>
</tr>
</tbody>
</table>

See more tables in the text, p.39!
2.2 Designing for performance

- Today’s computer basic building blocks are still the ones of the IAS!...

- But many “tricks” have been invented to improve performance
Some of the main “tricks”

- Pipelining
- On board cache
- On board L1 & L2 cache
- Branch prediction
- Data flow analysis
- Speculative execution
Performance Balance

- Processor speed increased
- Memory capacity increased
- Memory speed lags behind processor speed
  - See Fig. 2.10 on next slide
Logic and Memory Performance Gap

MHz


logic

memory
**Solutions**

- Increase number of bits retrieved at one time
  - Make DRAM “wider” rather than “deeper”
- Change DRAM interface
  - Cache
- Reduce frequency of memory access
  - More complex cache and cache on chip
- Increase interconnection bandwidth
  - High speed buses
  - Hierarchy of buses
I/O Devices

- Peripherals with intensive I/O demands
  - E.g. Computer clusters
- Large data throughput demands
- Processors can handle this
- Problem moving data
- Solutions:
  - Caching
  - Buffering
  - Higher-speed interconnection buses
  - More elaborate bus structures
    - E.g. InfiniBand
  - Multiple-processor configurations
Typical I/O Device Data Rates

- Gigabit Ethernet
- Graphics display
- Hard disk
- Ethernet
- Optical disk
- Scanner
- Laser printer
- Floppy disk
- Modem
- Mouse
- Keyboard
Key is Balance among:

- Processor components
- Main memory
- I/O devices
- Interconnection structures
To do for next time

• Read from text sections
  — 2.3 – Evolution of Intel x86
  — 2.4 – Embedded systems and the ARM

• Take notes in notebook!
Embedded Systems - ARM

- ARM evolved from RISC design
- Early 1980s: Acorn Computers (ARM = Acorn RISC Machine)
- Although initially intended for a general-use microcomputer, today it’s used mainly in **embedded systems**
  - Used within a larger product
  - Not a general-purpose computer
  - Dedicated function
  - E.g. Anti-lock brakes in car
Embedded Systems Requirements

- **Different sizes**
  - Different constraints, optimization, reuse

- **Different requirements**
  - Safety, reliability, real-time, flexibility, legislation
  - Lifespan
  - Environmental conditions
  - Static v dynamic loads
  - Slow to fast speeds
  - Computation-intensive vs. I/O-intensive
  - Discrete-event vs. continuous dynamics
Possible Organization of an Embedded System

- FPGA/ASIC
- Memory
- Auxiliary Systems (power, cooling)
- Processor
- Diagnostic port
- A/D conversion
- D/A Conversion
- Electromechanical backup and safety
- Sensors
- Actuators
- External environment
- Human interface
- Software
ARM

- Designed by ARM Inc., Cambridge, England
- It’s not a processor, but an architecture!
- AMR licenses it to manufacturers

- As of 2007, about 98 percent of the more than one billion mobile phones sold each year use at least one ARM processor
Improvements in Chip Organization and Architecture

The main goal/driver is the increase of CPU speed
How to increase CPU speed

1. Increase **hardware** speed of processor

How? Shrinking logic gate size

Pros:
- Smaller transistor $\rightarrow$ smaller capacitance $\rightarrow$
  - Smaller RC
  - Smaller current per gate
- Shorter wires $\rightarrow$ smaller R $\rightarrow$ smaller RC
- Gates closer together $\rightarrow$ shorter “speed-of-light” propagation delay
How to increase CPU speed

Shrinking logic gate size

How to increase CPU speed

1. Increase **hardware** speed of processor

How? Shrinking logic gate size

Cons:
- Connection wires are packed closer together $\rightarrow$ larger “parasitic” $C$ $\rightarrow$ larger $RC$
- Thinner wires $\rightarrow$ larger $R$ $\rightarrow$ larger $RC$
- Power dissipated by a switching CMOS gate is given by $f \cdot C \cdot V^2$ formula
How to increase CPU speed

1. Increase **hardware** speed of processor

Instead of a conclusion:

- [Intel sets the bar at 4GHz - CNET News, Nov. 2003](#)
- [Intel kills plans for 4GHz Pentium - CNET News, Oct 2004](#)
How to increase CPU speed

2. Increase size and speed of caches

How? Dedicating part of CPU chip to cache memory

“With the 486 processor, an 8 KB cache was integrated directly into the CPU die. This cache was termed Level 1 or L1 cache to differentiate it from the slower on-motherboard, or Level 2 (L2) cache.” Wikipedia
Increased Cache Capacity

- Typically two or three levels of cache between processor and main memory
- Chip density increased
  - More cache memory on chip
    - Faster cache access
- Pentium chip devoted about 10% of chip area to cache
- Pentium 4 devotes about 50%
How to increase CPU speed

3. Change processor organization and architecture
   — Increase effective speed of execution
   — Parallelism
More Complex Execution Logic

• Enable parallel execution of instructions

• Pipeline works like assembly line
  — Different stages of execution of different instructions at same time along pipeline

• Superscalar architecture allows multiple pipelines within a single processor
  — Instructions that do not depend on one another can be executed in parallel
Diminishing Returns

• Internal organization of processors is already very complex
  — Can get a great deal of parallelism
  — Further significant increases likely to be relatively modest

• Benefits from caches are reaching a limit

• Increasing clock rate runs into power dissipation problem
  — Some fundamental physical limits are being reached
New Approach – Multiple Cores

- Multiple processors on a single chip
  - Large shared cache
- Within a processor, increase in performance proportional to square root of increase in complexity
- If software can use multiple processors, doubling number of processors almost doubles performance
- So, use two simpler processors on the chip rather than one more complex processor
- With two processors, larger caches are justified
  - Power consumption of memory logic is less than processing logic

Why?
2.5 Assessment of computer performance

- Key parameters
  - Performance, cost, size, security, reliability, power consumption
- System clock speed
  - In Hz or multiples thereof
  - Clock rate, clock cycle, clock tick, cycle time
- Signals in CPU take time to settle down to 1 or 0
- Signals may change at different speeds
- Operations need to be synchronised
System Clock

Assessment of computer performance

- Instruction execution takes place in discrete steps
  - Fetch, decode, load and store, arithmetic or logical
  - Usually require multiple clock cycles per instruction
- Pipelining → simultaneous execution of instructions
- Conclusion: clock speed is not the whole story
Instruction Execution Rate

- Millions of instructions per second (MIPS)
- Millions of floating point instructions per second (MFLOPS)
- Heavily dependent on:
  - instruction set
  - compiler design
  - processor implementation
  - cache & memory hierarchy
Benchmarks

- Programs designed to test performance
- Written in high level languages
  - Portable
- Represents style of task
  - Systems, numerical, commercial
- Easily measured
- Widely distributed
- E.g. System Performance Evaluation Corporation (SPEC)
  - CPU2006 for computation bound
    - 17 floating point programs in C, C++, Fortran
    - 12 integer programs in C, C++
    - 3 million lines of code
  - Speed and rate metrics
    - Single task and throughput
SPEC Speed Metric

- Single task
- Base runtime defined for each benchmark using reference machine
- Results are reported as ratio of reference time to system run time
  - $T_{ref_i}$ execution time for benchmark $i$ on reference machine
  - $T_{syt_i}$ execution time of benchmark $i$ on test system

$$r_i = \frac{T_{ref_i}}{T_{syt_i}}$$

- Overall performance calculated by averaging ratios for all 12 integer benchmarks
  - Use geometric mean
    - Appropriate for normalized numbers such as ratios

$$R_G = \left( \prod_{i=1}^{n} r_i \right)^{1/n}$$
**SPEC Rate/Throughput Metric**

- Measures throughput or rate of a machine carrying out a number of tasks
- Multiple copies of benchmarks run simultaneously
  - Typically, same as number of processors
- Ratio is calculated as follows:
  - $T_{ref_i}$ is the reference execution time for benchmark $i$
  - $N$ is the number of copies run simultaneously
  - $T_{sut_i}$ the time elapsed from start of execution of program on all $N$ processors until completion of all copies of program
  - Again, a geometric mean is calculated

\[
r_i = \frac{N \times T_{ref_i}}{T_{sut_i}}
\]
Amdahl’s Law

- Gene Amdahl [AMDA67]
- Potential speed-up of program using multiple processors
- Concluded that:
  - Code needs to be parallelizable
  - Speed up is bound, giving diminishing returns for more processors
- Task dependent
  - Servers gain by maintaining multiple connections on multiple processors
  - Databases can be split into parallel tasks
Amdahl’s Law

For a program running on a single processor

- Fraction $f$ of code infinitely parallelizable with no scheduling overhead
- Fraction $(1-f)$ of code inherently serial
- $T$ is total execution time for program on single processor
- $N$ is number of processors that fully exploit parallel portions of code

$$Speedup = \frac{\text{time to execute program on a single processor}}{\text{time to execute program on } N \text{ parallel processors}} = \frac{T(1-f) + Tf}{T(1-f) + \frac{Tf}{N}} = \frac{1}{(1-f) + \frac{f}{N}}$$
Amdahl’s Law

\[
\text{Speedup} = \frac{\text{time to execute program on a single processor}}{\text{time to execute program on } N \text{ parallel processors}} = \frac{T(1-f) + Tf}{T(1-f) + \frac{Tf}{N}} = \frac{1}{(1-f) + \frac{f}{N}}
\]

Conclusions

- \( N \to \infty \), speedup bound by \( 1/(1 - f) \)
- \( f \) small \( \rightarrow \) parallel processors has little effect
  - Diminishing returns for using more processors
Homework #1 (for Ch.2), due next Thu

Solve the following end-of-chapter problems:

- **2.1**
  - We are explaining loops in the lab today!
- **2.2**
  - Refer to figure 2.3 on p.22 of text
- **2.3**
  - Refer to figure 2.3 on p.22 of text
- **2.6**
- **2.10**
- **2.13**