Lecture 1: Course introduction

- Course organization
- Historical overview
- Computer organization
- Why the MC68000?
- Why assembly language?
Course organization

- **Grading**
  - Exams
    - 1 midterm and 1 final
  - Homework
    - 4 problem sets (not graded)
  - Quizzes
    - Biweekly
  - Laboratories
    - 5 Labs

- **Grading scheme**

<table>
<thead>
<tr>
<th></th>
<th>Weight (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Quizzes</td>
<td>20</td>
</tr>
<tr>
<td>Laboratory</td>
<td>40</td>
</tr>
<tr>
<td>Midterm</td>
<td>20</td>
</tr>
<tr>
<td>Final Exam</td>
<td>20</td>
</tr>
</tbody>
</table>
**Course outline**

- **Module I: Programming (8 lectures)**
  - MC68000 architecture (2)
  - Assembly language (5)
    - Instruction and addressing modes (2)
    - Program control (1)
    - Subroutines (2)
  - C language (1)

- **Module II: Peripherals (9)**
  - Exception processing (1)
  - Devices (6)
    - PI/T timer (2)
    - PI/T parallel port (2)
    - DUART serial port (1)
  - Memory and I/O interface (1)
  - Address decoding (2)
### Brief History of Computers

<table>
<thead>
<tr>
<th>GENERATION</th>
<th>FEATURES</th>
<th>MILESTONES</th>
<th>YEAR</th>
<th>NOTES</th>
</tr>
</thead>
<tbody>
<tr>
<td>Early machines</td>
<td>Mech., Electro-</td>
<td>Asia Minor, Abacus</td>
<td>3000BC</td>
<td>Only replaced by paper and pencil</td>
</tr>
<tr>
<td>(3000BC-1945)</td>
<td>mech.</td>
<td>Blaise Pascal, Pascaline</td>
<td>1642</td>
<td>Decimal addition (8 decimal figs)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Charles Babbage</td>
<td>1823</td>
<td>Steam powered</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Differential Engine</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Herman Hollerith</td>
<td>1889</td>
<td>US census, origin of IBM</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Howard Aiken, Harvard</td>
<td>1937</td>
<td>Ballistic charts of US Navy</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Mark I</td>
<td></td>
<td></td>
</tr>
<tr>
<td>First</td>
<td>Vacuum tubes</td>
<td>Alan Turing, Colossus</td>
<td>1943</td>
<td>Decode German ENIGMA codes</td>
</tr>
<tr>
<td>(1945-1956)</td>
<td></td>
<td>Eckert, Mauchly, ENIAC</td>
<td>1946</td>
<td>1st general purpose electronic computer</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Von Neumann, EDVAC</td>
<td>1950</td>
<td>Von Neumann architecture</td>
</tr>
<tr>
<td>Second</td>
<td>Transistor (1947)</td>
<td>MIT Lincoln Labs, TXO</td>
<td>1953</td>
<td>1st computer based on transistors</td>
</tr>
<tr>
<td>(1956-1963)</td>
<td></td>
<td>High level programming languages</td>
<td>1956</td>
<td>FORTRAN (1956), COBOL (1959)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IBM Stretch, Sperry-Rand LARC</td>
<td>1950s</td>
<td>1st supercomputers, scientific computation</td>
</tr>
<tr>
<td>Third</td>
<td>IC (SSI, MSI)</td>
<td>Seymour Cray, CDC 6600</td>
<td>1964</td>
<td>1st to use parallelism (10 processors)</td>
</tr>
<tr>
<td>(1964-1971)</td>
<td></td>
<td>IBM SYSTEM 360</td>
<td>1964</td>
<td>Makes other systems obsolete</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DEC PDP-8</td>
<td>1965</td>
<td>1st successful minicomputer</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Intel 4004</td>
<td>1971</td>
<td>4-bit (1st microprocessor)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Intel 8008</td>
<td>1972</td>
<td>8/8/14</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Motorola 6800</td>
<td>1974</td>
<td>8/8/16</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Intel 8086</td>
<td>1978</td>
<td>16/16/20</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Motorola 68000</strong></td>
<td>1979</td>
<td><strong>32/16/24</strong></td>
</tr>
<tr>
<td></td>
<td>Micro-processor</td>
<td>Intel 80286</td>
<td>1982</td>
<td>16/16/24</td>
</tr>
<tr>
<td>Fourth</td>
<td>LSI, VLSI</td>
<td>Motorola 68020</td>
<td>1984</td>
<td>32/32/32</td>
</tr>
<tr>
<td>(1971-present)</td>
<td></td>
<td>Intel 80386</td>
<td>1985</td>
<td>32/32/32, pipelining</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Motorola 68030</td>
<td>1987</td>
<td>32/32/32, MMU</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Intel 80486</td>
<td>1989</td>
<td>32/32/32, cache, FPP</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Motorola 68040</td>
<td>1991</td>
<td>32/32/32, FPP</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Motorola Power PC 601</td>
<td>1993</td>
<td>32/64/32, RISC, super-scalar</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(G1)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Motorola Power PC 603</td>
<td>1994</td>
<td>32/64/32, super-scalar</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(G2)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Motorola Power PC 604</td>
<td>1994</td>
<td>32/64/32, portable computing</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(G3)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Intel Pentium</strong></td>
<td>1993</td>
<td>32/64/32, super-scalar</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Intel Pentium Pro</strong></td>
<td>1995</td>
<td>32/64/32 (optimized for 32-bit OS)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Motorola Power PC 620</strong></td>
<td>1996</td>
<td>64/64/32</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(G4)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Intel Pentium II</strong></td>
<td>1997</td>
<td>32/64/32, MMX</td>
</tr>
</tbody>
</table>
Anecdote: the first ‘bug’ (1945)

- Grace Murray Hopper, working in a temporary World War I building at Harvard University on the Mark II computer, found the first computer bug (a moth) beaten to death in the jaws of a relay.
  - She glued it into the logbook of the computer and thereafter when the machine stopped (frequently) they would tell Howard Aiken that they were "debugging" the computer.
  - The very first bug still exists in the National Museum of American History of the Smithsonian Institution. The word bug and the concept of debugging had been used previously, perhaps by Edison, but this was probably the first verification that the concept applied to computers.
Computer organization building blocks

- **Memory**
  - Stores the instructions and data comprising the program to be executed

- **CPU**
  - Interprets and executes program instructions in sequence

- **I/O devices**
  - Communicates the CPU with the *real world*

- **System buses**
  - A collection of wires that allow access to the circuitry around the CPU

![Diagram showing CPU, Memory, and I/O device connections with buses](image)
Memory types

- **Classification by access**
  - **Registers**: Internal to the CPU
  - **Cache**: Tightly linked to the CPU
  - **Main memory**: Connected to system buses
  - **Secondary memory**: On-line
  - **Archival**: Off-line

- **Classified by technology**
  - **Semiconductor**
    - Random Access Memory (RAM)
      - Volatile, Read and write
    - Read Only Memory (ROM)
      - Non-volatile, Can only be read
  - **Magnetic memory**
    - Disk, Tape
  - **Optical**
    - CD-ROM

<table>
<thead>
<tr>
<th>MEMORY TECHNOLOGY</th>
<th>RAM</th>
<th>ROM</th>
<th>Disk</th>
<th>Tape</th>
<th>Optical</th>
</tr>
</thead>
<tbody>
<tr>
<td>Registers</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cache</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Main memory</td>
<td>X</td>
<td>X</td>
<td>Virtual</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Secondary memory</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Archival storage</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

[Diagram showing memory types and access]

- Fast and Expensive
- Cheap and Slow
Central Processing Unit

- **Arithmetic-Logic Unit (ALU)**
  - Performs the operations required by the CPU

- **Control Unit**
  - Determines the operation to be performed by an instruction
  - Sets in motion necessary actions to perform the operation

- **Registers**
  - Data/Address Registers
  - Instruction Register
  - Program Counter
  - Status Register
  - Memory Data/Address Registers
I/O devices

- **Mass-Storage Devices**
  - Hold large quantities of information that cannot fit into the computer’s main memory
  - Disks, tapes, CD-ROMs

- **Human Interface Devices**
  - Input: keyboard, mouse, ...
  - Output: displays, printers, ...

- **Control/Monitor Devices**
  - Control devices are actuators (outputs)
  - Monitor devices are sensors (inputs)
System buses

- **Address Bus**
  - Carries the location in memory of a given item
  - Uni-directional (always supplied by the CPU)
  - Determines maximum amount of memory available to CPU

- **Data bus**
  - Carries data between CPU and memory or I/O devices
  - Bi-directional
  - Determines the width of the architecture

- **Control Bus**
  - Carries timing signals (and more) to synchronize CPU to external circuitry
  - Highly dependent on the specific CPU
CPU operation

- CPU “fetch-execute” cycle
  - fetch instruction from memory
  - decode instruction
  - perform operations required by the instruction

function von_newmann {
    pc=init_pc();
    while (not_done) {
        opcode = fetch_instr(memory[pc]);
        execute(opcode);
        pc=pc+1;
    }
}

function execute(opcode) {
    decop = decode(opcode);
    if need_data(decop) {
        data = get_data(decop);
    }
    result = compute(decop, data);
    if save_result(decop) {
        save_result(decop, result);
    }
}
Opcode fetch

CPU

Program Counter

opcode

Instruction Register

MEMORY

opcode

N

N+1

N

N-1

Address bus

Data bus

Clock

Read

Control bus
Why the MC68000?

- **A straightforward and ‘nice’ μP**
  - Powerful and relatively simple instruction set
  - Sophisticated interfacing capabilities
  - Ability to support multi-tasking
  - The most popular μP family in academia
  - Flat memory map

Look at the microcontroller market
Why assembly language?

- **Hardware prospective**
  - Assembly language teaches how a computer works at the machine level (i.e. registers)
  - Assembly language helps understand the limitations of the Von Neumann architecture

- **Software prospective**
  - The foundation of many abstract issues in software lies in assembly language and computer architecture:
    - Data types, addressing modes, stack, recursion, input/output

- **Assembly language is not used just to illustrate algorithms, but to demonstrate what is actually happening inside the computer!**
Micro-processor Vs. Micro-controller

- **Micro-processor (MPU, μP)**
  - CPU alone
  - may contain some memory
  - classified by data path width 4, 8, 16, 32 or 64 bits
  - Ex: MC68000

- **Micro-controller (MCU)**
  - microprocessor plus peripherals on a single chip
  - a one chip computer system
  - additional peripherals may be interfaced separately
  - Ex: 68HC11