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

	Weight (%)
Quizes	20
Laboratory	40
Midterm	20
Final Exam	20

Instructor

Ricardo Gutierrez-Osuna
Office: 401 Russ
Tel:775-5120
rgutier@cs.wright.edu
http://www.cs.wright.edu/~rgutier
Office hours: TBA

Teaching Assistant

Mohammed Tabrez
Office: 339 Russ
tmohamme@cs.wright.edu
Office hours: TBA

Course outline

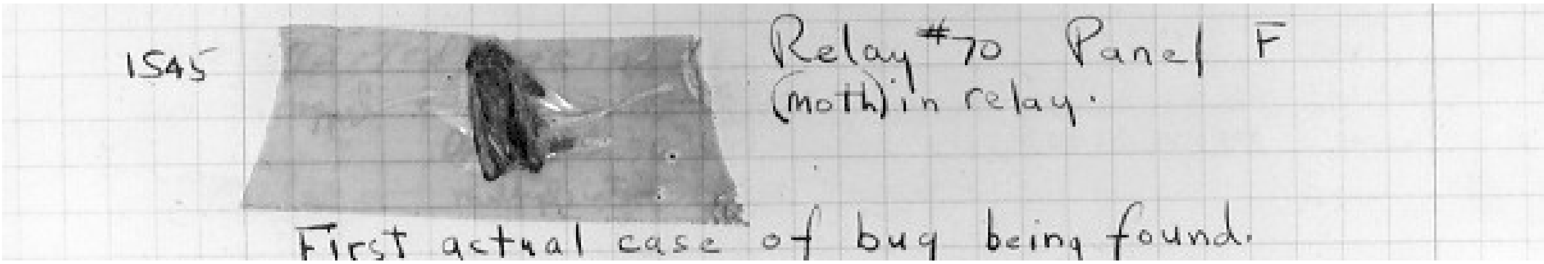
- **Module I: Programming (8 lectures)**
  - MC68000 architecture (2)
  - Assembly language (5)
    - Instruction and addressing modes (2)
    - Program control (1)
    - Subroutines (2)
  - C language (1)
- **Module II: Peripherals (9)**
  - Exception processing (1)
  - Devices (6)
    - PI/T timer (2)
    - PI/T parallel port (2)
    - DUART serial port (1)
  - Memory and I/O interface (1)
  - Address decoding (2)

Brief history of computers

GENERATION	FEATURES	MILESTONES	YEAR	NOTES
Early machines (3000BC-1945)	Mech., Electro-mech.	Asia Minor, Abacus	3000BC	Only replaced by paper and pencil
		Blaise Pascal, Pascaline	1642	Decimal addition (8 decimal figs)
		Charles Babbage Differential Engine	1823	Steam powered
		Herman Hollerith, Punch Card	1889	US census, origin of IBM
		Howard Aiken, Harvard Mark I	1937	Ballistic charts of US Navy
First (1945-1956)	Vacuum tubes	Alan Turing, Colossus	1943	Decode German ENIGMA codes
		Eckert, Mauchly, ENIAC	1946	1st general purpose electronic computer
		Von Neumann, EDVAC	1950	Von Neumann architecture
Second (1956-1963)	Transistor (1947)	MIT Lincoln Labs, TXO	1953	1st computer based on transistors
		High level programming languages	1956	FORTRAN (1956), COBOL (1959)
		IBM Stretch, Sperry-Rand LARC	1950s	1st supercomputers, scientific computation
Third (1964-1971)	IC (SSI, MSI)	Seymour Cray, CDC 6600	1964	1st to use parallelism (10 processors)
		IBM SYSTEM 360	1964	Makes other systems obsolete
		DEC PDP-8	1965	1st successful minicomputer
Fourth (1971-present)	Micro-processor LSI, VLSI	Intel 4004	1971	4-bit (1st microprocessor)
		Intel 8008	1972	8/8/14
		Motorola 6800	1974	8/8/16
		Intel 8086	1978	16/16/20
		Motorola 68000	**1979**	**32/16/24**
		Intel 80286	1982	16/16/24
		Motorola 68020	1984	32/32/32
		Intel 80386	1985	32/32/32, pipelining
		Motorola 68030	1987	32/32/32, MMU
		Intel 80486	1989	32/32/32, cache, FPP
		Motorola 68040	1991	32/32/32, FPP
		Motorola Power PC 601 (G1)	1993	32/64/32, RISC, super-scalar
		Intel Pentium	1993	32/64/32, super-scalar
		Motorola 68060	1994	32/32/32, super-scalar
		Motorola Power PC 603 (G2)	1994	32/64/32, portable computing
		Motorola Power PC 604 (G3)	1994	32/64/32, server, workstations
		Intel Pentium Pro	1995	32/64/32 (optimized for 32-bit OS)
		Motorola Power PC 620 (G4)	1996	64/64/32
		Intel Pentium II	1997	32/64/32, MMX

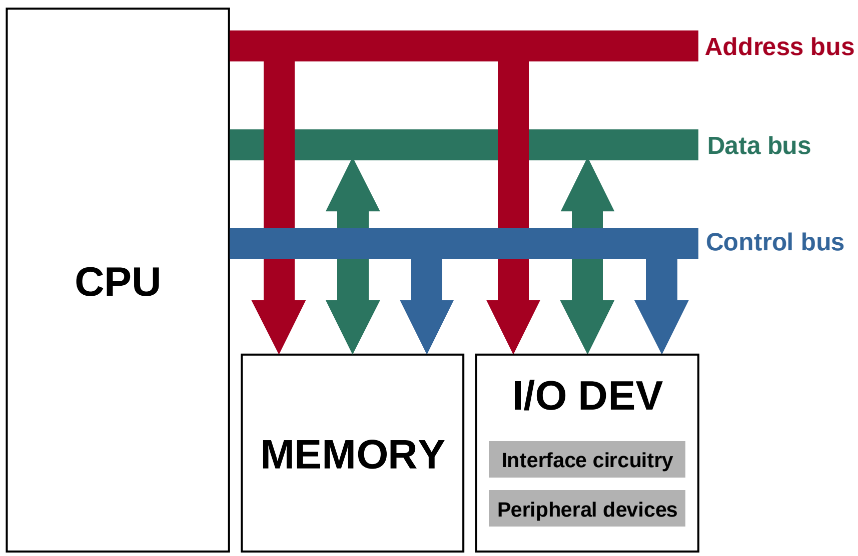
Anecdote: the first 'bug' (1945)

- **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.



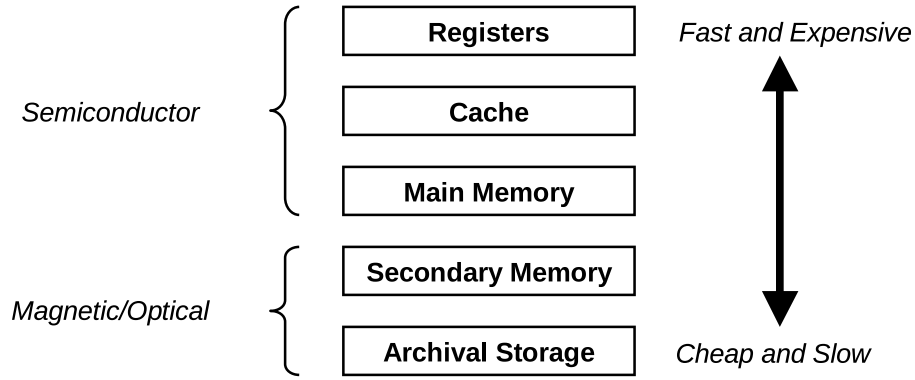
Computer organization building blocks

- **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

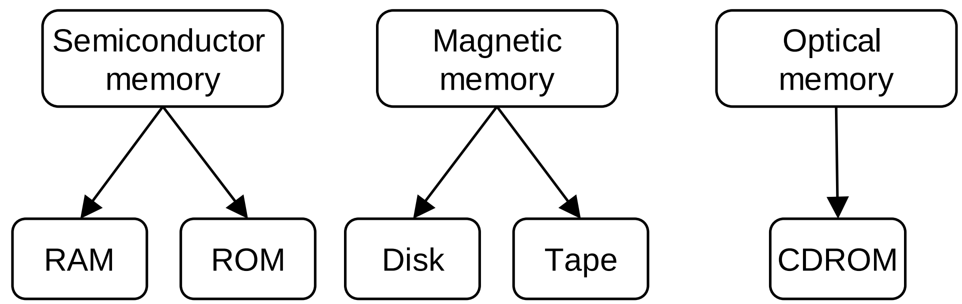


Memory types

- **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



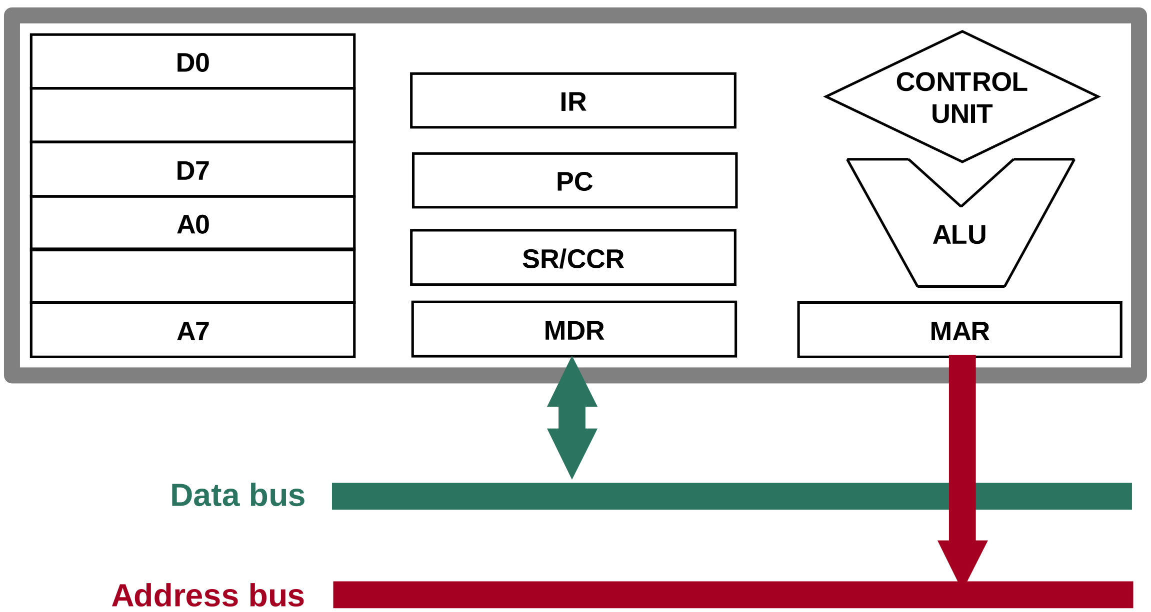
- **Classified by technology**
  - Semiconductor
    - Random Access Memory (RAM)
      - Volatile, Read and write
    - Read Only Memory (ROM)
      - Non-volatile, Can only be read
  - Magnetic memory
    - Disk, Tape
  - Optical
    - CD-ROM



	MEMORY TECHNOLOGY				
MEMORY ACCESS	***RAM***	***ROM***	***Disk***	***Tape***	***Optical***
Registers	X				
Cache	X				
Main memory	X	X	Virtual		
Secondary memory			X		X
Archival storage				X	X

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

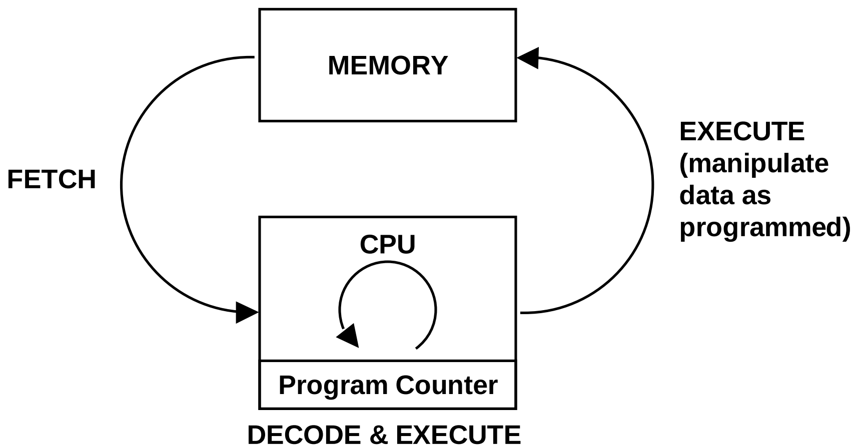
- **Address Bus**
  - Carries the location in memory of a given item
  - Uni-directional (always supplied by the CPU)
  - Determines maximum amount of memory available to CPU
- **Data bus**
  - Carries data between CPU and memory or I/O devices
  - Bi-directional
  - Determines the width of the architecture
- **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

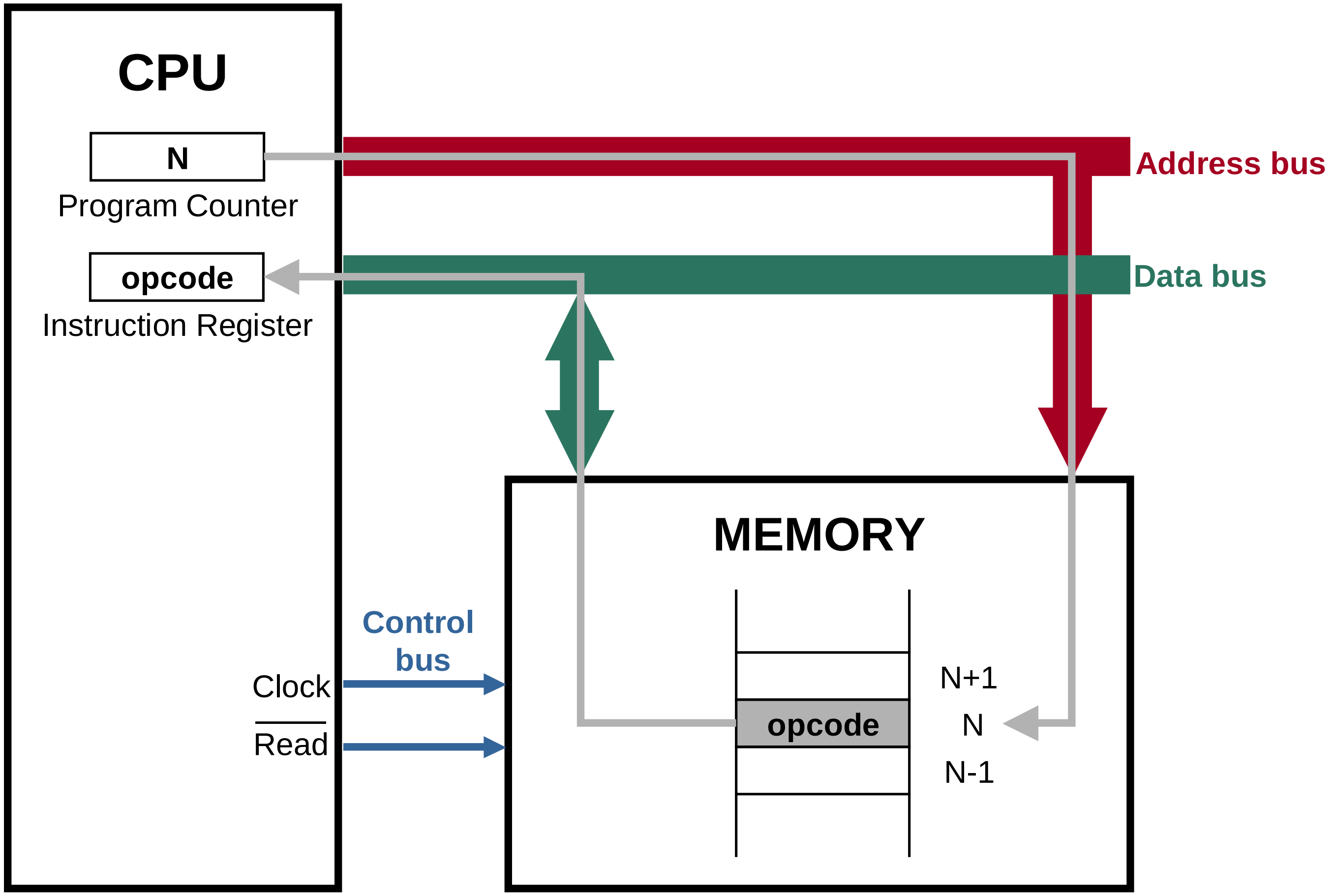
- fetch instruction from memory
- decode instruction
- perform operations required by the instruction

```
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

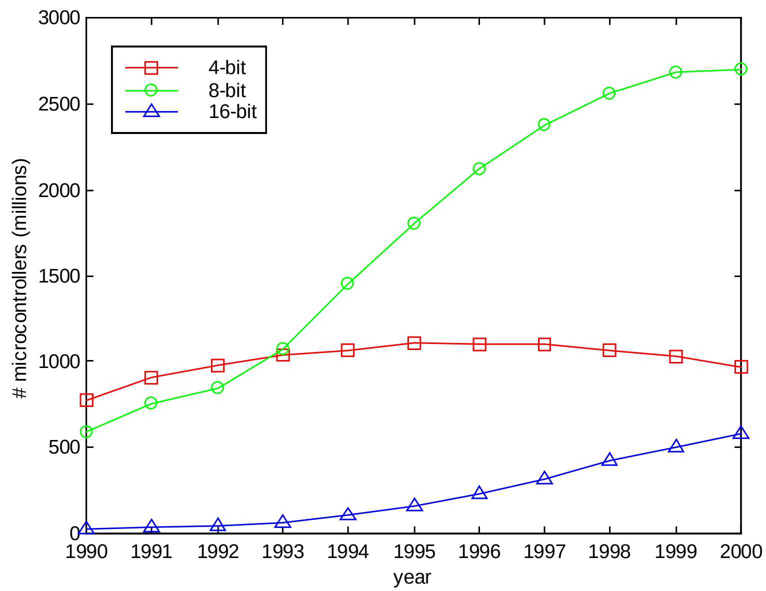


Why the MC68000?

■ 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 micro-controller 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